NAVAL POSTGRADUATE SCHOOL
Monterey, California

THESIS

**FREE- FIELD SPATIALIZED
AURAL CUES FOR
SYNTHETIC ENVIRONMENTS**

by

John T. Roesli

September 1994

Thesis Advisor: Michael M. Zyda
Thesis Co-Advisor: John S. Falby

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY (Leave Blank)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	September 1994	Master's Thesis

4. TITLE AND SUBTITLE
FREE-FIELD SPATIALIZED AURAL CUES FOR SYNTHETIC ENVIRONMENTS(U)

5. FUNDING NUMBERS

6. AUTHOR(S)
Roesli, John T.

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Naval Postgraduate School
Monterey, CA 93943-5000

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING/ MONITORING AGENCY NAME(S) AND ADDRESS(ES)

10. SPONSORING/ MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES
The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the United States Government.

12a. DISTRIBUTION / AVAILABILITY STATEMENT
Approved for public release; distribution is unlimited.

12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*

Commercially available spatial audio systems for synthetic environments suffer from excessive cost and the requirement for in-house application software development. The purpose of this work was to develop a low cost audio hardware and software system capable of generating aural cues for a synthetic environment in real-time, which correctly reflects the user's location and accurately conveys the type and location of the sound event.

The approach taken was to first implement a software communication package using DIS (Distributed Interactive Simulation protocol, a Department of Defense standard) to retrieve information from the virtual world. The second step was to develop algorithms and software to process that information and model the physical sound world. Finally, an audio hardware system capable of generating the required audio cues in real-time was constructed.

The result of this work is a system consisting of software and audio hardware for generating spatial aural cues that correctly localize a sound event for users in a virtual world. The system makes use of "off-the-shelf" audio hardware (MIDI capable sampler, amplifiers, and speakers) which reduces the cost from $20,000 to less than $5,000. With minor modifications for MIDI port access and graphics library function calls, the software can be utilized on any computer that reads DIS packets from the network and writes MIDI data to a data port.

14. SUBJECT TERMS	15. NUMBER OF PAGES
MIDI, Spatial Audio, DIS, Simulation, Virtual World, Virtual Environment Sound, Audio, Surround Sound	100
	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
Unclassified	Unclassified	Unclassified	UL

Approved for public release; distribution is unlimited

FREE FIELD SPATIALIZED AURAL CUES FOR SYNTHETIC ENVIRONMENTS

by

John T. Roesli
Lieutenant, United States Navy
B.S., University Nevada Reno, 1985

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN COMPUTER SCIENCE

from the

NAVAL POSTGRADUATE SCHOOL

September 1994

Author:

_____

John T. Roesli

Approved By:

_____

Michael M. Zyda, Thesis Advisor

_____

John S. Falby, Thesis Co-Advisor

_____

Ted Lewis, Chairman,
Department of Computer Science

ABSTRACT

Commercially available spatial audio systems for synthetic environments suffer from excessive cost and the requirement for in-house application software development. The purpose of this work was to develop a low cost audio hardware and software system capable of generating aural cues for a synthetic environment in real-time, which correctly reflects the user's location and accurately conveys the type and location of the sound event.

The approach taken was to first implement a software communication package using DIS (Distributed Interactive Simulation protocol, a Department of Defense standard) to retrieve information from the virtual world. The second step was to develop algorithms and software to process that information and model the physical sound world. Finally, an audio hardware system capable of generating the required audio cues in real-time was constructed.

The result of this work is a system consisting of software and audio hardware for generating spatial aural cues that correctly localize a sound event for users in a virtual world. The system makes use of "off-the-shelf" audio hardware (MIDI capable sampler, amplifiers, and speakers) which reduces the cost from $20,000 to less than $5,000. With minor modifications for MIDI port access and graphics library function calls, the software can be utilized on any computer that reads DIS packets from the network and writes MIDI data to a data port.

ACKNOWLEDGEMENTS

This work is the culmination of two years of programming and research. Along the way many people have contributed significantly to the development of the source code and hardware configuration. I would like to first of thank Susannah Bloch, her programming assistance with the original sound server made the project possible much sooner than originally anticipated. John Locke unknowingly played a major by development of the network code. I would also like to thank Dave Young and Dr. David Pratt, their assistance with the initial data extraction idea really got the project started. Richard Hashimoto and his assistance in using the spectral analyzer for testing of spatial and loudness algorithms. Also, I would like to thank Paul Barham for his help in developing the host logic, data structures and graphic display. Last but not least thanks to John Falby and Dr. Michael Zyda for their continuous support and assistance.

Table of Contents

LIST OF FIGURES

I. INTRODUCTION

A. SPATIAL AUDIO

In recent years, the increasing ability of low-end graphics workstations to produce quality images at usable frame rates for real-time display has enabled a multitude of commercial and research institutions to begin exploring virtual environments. Some applications are as simple as improving the quality of arcade style games, while others include military war gaming and simulation, scientific visualizations, and telepresent robotic applications. Along with these new and improved visualization tools, a ground swell of interest in three dimensional audio, also referred to as spatial audio, has emerged. In a virtual environment, as you move to the left or right, you expect the view to move accordingly. Similarly, if an audio event occurs on your left, such as a ball impacting a wall, you expect to hear the sound to your left.

B. QUESTIONS

In designing an audio system, the search began with a look at commercial products available to meet the requirements. It became apparent early on in research and development that there were very few commercial products available to meet these requirements. The products that were available entail an exorbitant cost and are thus prohibitive.

This lead to designing a system using existing hardware and computing power available in the lab. Once the design process had begun a multitude of questions were raised. How do you get information from the virtual world, in order to determine what kind of aural cues to generate, when to generate them, and where to generate them? How to generate the audio cues? What type of delivery system, i.e. Free-Field (external audio speakers) or headphones, and what effects the system will have on the listener?

C. APPLICATIONS

A spatial audio system has a myriad of applications. For example, pilots are increasingly overloaded with visual information in a cockpit. If information like an ESM warning were spatialized, the pilot would not only here the warning, but the sound would be heard from the bearing of the threat.

An example of a shipboard application would be in the engineering plant. The control console for the main engineering plant of a typical cruiser or destroyer is an enormous panel of lights and switches. The warning buzzer is a one inch diameter speaker in the lower left hand corner. When an alarm occurs the watchstander first reaches for a button to cancel the sound of the alarm, then searches the control panel to see what light is flashing, then begins to take appropriate action. If the information were conveyed in the form of spatial audio, the watchstander would instantly know the area of the console to look at and this would result in decreased response time and greater ease in determining what problem had occurred.

The addition of spatial audio cues to a synthetic environment dramatically increases the level of immersion for the user. At the Naval Postgraduate School, we began adding audio cues to virtual environments in 1989, simply playing hard coded sound files from a Macintosh PC. After observing the benefits gained from adding very basic audio cues, our goal became to develop a method of creating real-time spatial audio cues related to geographic and event driven scenarios in the virtual environment.

II. BACKGROUND

This chapter provides a brief background of sound localization and MIDI principles necessary to understand the method of spatial audio that has been developed in the Naval Postgraduate School's Graphics and Video laboratory.

A. SOUND LOCALIZATION

Sound can be defined as a localized change in pressure that causes compression and refraction through a medium. This can be characterized as a wave. As a wave it has amplitude, frequency, velocity, and time. Amplitude is perceived as loudness and frequency is perceived as pitch. Velocity is characterized by the compliance of the medium in which the wave travels.

Perhaps the most common application of spatial audio is found in stereo recordings. The basic idea of a stereo recording is to place two microphones in a room, assign each microphone to a separate audio channel and record the sound. This results in capturing the differences in intensity and some phase difference between various points in the sound field [Ref. 1]. This form of recording and playback provides a sense of movement from left to right, but is limited to this horizontal axis, whether played back through headphones or external speakers.

1. Duplex Theory

Humans determine the locality of a sound based upon several factors. The "duplex theory" suggests two primary cues for sound localization [Ref. 2]. They are the ITD (Interaural Time Difference), which is the delay experienced when a sound reaches one ear before the other, and the IID (Interaural Intensity Difference), which is mainly caused by head-shadowing (see Figure 1). Although this explains some types of sound localization, there are several shortcomings of the duplex theory.

The duplex theory cannot account for the ability of subjects to localize sounds on the vertical median plane where interaural cues are minimal. Similarly, when subjects listen to stimuli over headphones, they are perceived as being inside the head even

though interaural temporal and intensity differences appropriate to an external source location are present. [Ref. 3]

2. Head Related Transfer Function

Current theory now suggests that the interaction between the inner and outer ear or pinnae provides a spectral shaping and is highly direction dependent [Ref. 3]. The absence of such cues severely degrades localization correctness [Ref. 4]. The pinna cues are chiefly responsible for the ability to externalize: the “outside-the-head” sensation [Ref. 5].

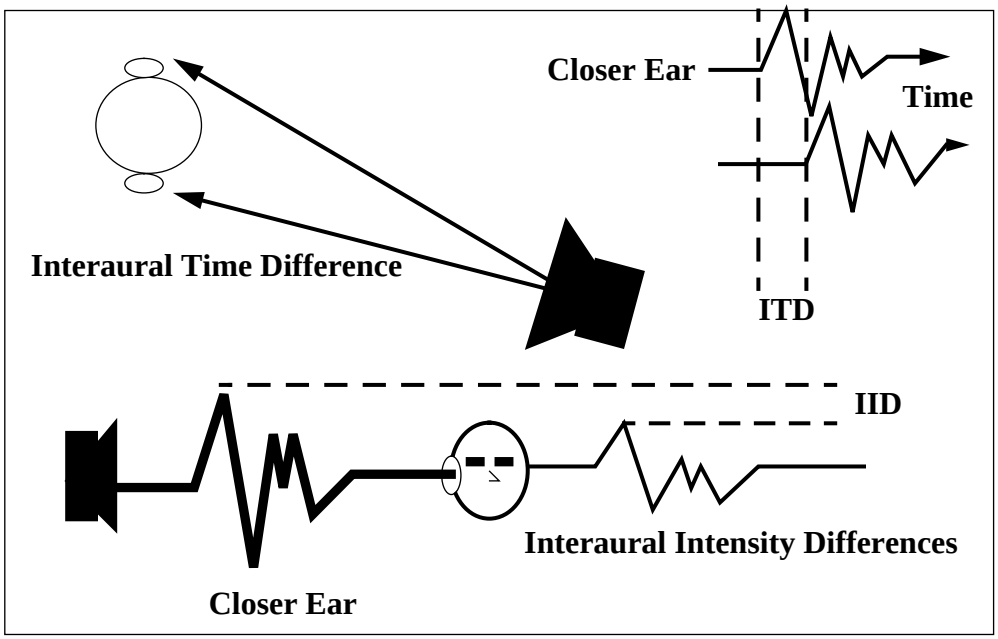


Figure 1: Two primary cues of sound localization [Ref. 6]

A method of recreating these effects is to capture the sum of all aspects affecting localization into a filter that can be applied to a sound. The aspects affecting localization can be captured by placing tiny microphones in a listeners ear, referred to as biaural recording, and producing a short sound pulse. The output of the microphones can be measured and used to create such a filter. The advantage to this method is that it captures the aggregate spatial cues for a particular source location, listener, and environment. These filters are considered finite impulse responses (FIR) and are referred to as a head-related transfer function (HRTF). By applying this filter to a given sound source, the spatial

location of the original filter can be recreated [Ref. 6]. The HRTF filters have provided a fairly accurate model of sound localization, however they are not without problems. Resolution of about 5 to 20 degrees is about the best that has been achieved. Blauert refers to this as localization blur [Ref. 7]. Back-to-front confusion [Ref. 8] and elevation confusion [Ref. 3] are also present. Reasons for this are not yet totally understood. One explanation is the so-called cone of confusion [Ref. 9], sounds emanating from certain bearings produce the same ITD's and IID's. There are many other problems associated with determining exactly how humans localize sound. Further information can be obtained from any of the references previously listed.

B. MIDI

Other than electronic musicians and a few hobbyists, the Musical Instrument Digital Interface (MIDI) is perhaps one of the most misunderstood protocols in use today [Ref. 10]. Often, perceptions are of some type of sound file system, when it actually has nothing to do with the actual digital sound files.

The best way to understand MIDI is to think of it as a communications protocol. A MIDI message is nothing more than a series of bytes sent to a synthesizer or sampler that conveys a multitude of commands.

1. Communication standard

MIDI communication [Ref. 11] is achieved through use of a 5-pin Deutsche Industrie Norm (DIN) socket. This cable connects a computer to a synthesizer or sample player. It is considered an asynchronous serial connection and communicates at a baud rate of 31.25 Kbaud (+/- 1%). This is a "non-standard" specification and prevents standard computer serial ports from working as a direct connection. Generally, an internal MIDI card or converter between the computer serial line and the MIDI device is required. Amiga is one of the few computer manufactures that provides a MIDI port standard with their computers.

2. Basic Message structure

The message structure is in the form of one device as transmitter and the other as receiver. There is no concept of an acknowledgment or handshaking mechanism. However, most machines will return some type of error message upon receipt of an erroneous message [Ref. 11].

A typical MIDI message consist of a status byte followed by 1 or 2 data bytes. Table 1 show a listing of what are called voice messages.

Status Byte	Meaning	Data Byte 1	Meaning	Data Byte 2	Meaning
0x80 - 0x8f	Note off	0x00 - 0x7f	Pitch	0x00 - 0x7f	Velocity
0x90 - 0x9f	Note on	0x00 - 0x7f	Pitch	0x00 - 0x7f	Velocity
0xa0 - 0xaf	Key pressure	0x00 - 0x7f	Pitch	0x00 - 0x7f	Pressure
0xb0 - 0xbf	Parameter	0x00 - 0x7f	Parameter number	0x00 - 0x7f	Setting
0xc0 - 0xcf	Program	0x00 - 0x7f	Program Selected	Not Used	Not Used
0xd0 - 0xdf	Channel Pressure	0x00 - 0x7f	Channel After Touch	Not Used	Not Used
0xe0 - 0xef	Pitch Wheel	0x00 - 0x7f	LSB	0x00 - 0x7f	MSB

Table 1: Layout of general MIDI commands

In MIDI the term velocity is synonymous with volume. The idea is that the velocity setting corresponds to the degree of force that should be applied to the keyboard. Pitch refers to the note that should be stuck e.g. middle C. Pressure refers to what is often called after touch. This is the degree to which pressure is applied to a key after the key was struck. After touch can be programmed to affect various parameters e.g. pan, filter cutoff, cross-fade. The message can be sent to a specific note or applied to an entire channel. Program refers to changes in instrument settings. To change from a piano to an organ all that is required is the proper program change for that particular keyboard. One other item to point out is that all of the status bytes have a range of sixteen. This corresponds to the sixteen

MIDI channels available. For data bytes the range is from 0 - 127. This seems to be a fairly sufficient range in that a standard piano has 88 notes, so we can actually play notes that are above and below this range. However, when using a 0 - 127 range for other parameters such as pitch wheel bending range or velocity it does cause some compromise in the number of steps increased or decreased, referred to as resolution.

The protocol was developed in 1983 and still has a long way to go in improving its capabilities, but the advantages are numerous. An entire musical score can be stored on a computer using less than 8K of disk space. The samples are stored in the sample player or digital keyboard and played by the MIDI file. To store the same musical file on a computer in one of the various digital sound formats could easily occupy 90 megabytes of disk space.

III. PREVIOUS WORK

A great deal of work has been done in the area of spatial audio. In recent years technology has provided hardware capable of processing digital sound in real-time at computational levels not even thought of 15 years ago. To enumerate all of the work done would fill volumes. The examples listed are those that contributed to the development of the system in use at the Naval Postgraduate School.

A. NASA AMES

A great deal of work has been done at NASA Ames Research Center involving the Head Related Transfer Functions [Ref. 3]. This work has used hardware designed by Scott Foster of Crystal River Engineering. In his system, a map of corrected HTRF filters have been downloaded from a host computer to the dual port memory of a real-time digital signal processor known as the Convolvotron.

> A set of two printed circuit boards converts one or more monaural analog inputs to digital signals at a rate of 50kHz (16-bit resolution). Each stream is then convolved with filter coefficients determined by the coordinates of the desired target locations and the position of the listeners head, thus placing each input signal in the perceptual 3-space of the listener. The resulting data streams are mixed, converted to left and right analog signals, and presented over head phones.[Ref. 3]

The main application for this work has been to create an auditory display for an aircraft pilot. This spatial auditory display gives the pilot the ability to interpret voice commands from multiple radio sources more accurately, and can also be used to provide aural cues that inform the pilot of the status of various aircraft flight and weapons systems.

1. Communication

The term “increased sensitivity for communication” [Ref. 6] refers to studies that have shown people to have the ability to separate an individual voice from many people speaking simultaneously in the same room. This effect is commonly referred to as the “cocktail party effect” [Ref. 1]. When a monaural input of multiple voices is delivered to an individual wearing headphones, the ability to separate out the voices and focus on a

particular speaker is very poor. When the same four voices are presented to the listener with a spatial locality for each voice, the listeners ability to focus on individual voices improves significantly [Ref. 6].

2. Auditory Space

Another application of spatial audio is in the ability to create a artificial aural environment. Referred to as the "minds aural eye" [Ref. 6], this can be used in three ways.

a. Urgency

When all alarms are of the same loudness and are presented from one single source in the cockpit, the pilot has no way to determine the urgency of the alarm. By providing spatial locality, variance in loudness, or altering the pitch of the alarm the pilot can immediately discern the implication of the alarm as well as a location to look for further information or responses that may be required [Ref. 6].

b. Redundancy

Another form of audio imagery is through the use of "Auditory icons and redundancy" [Ref. 6]. This can assist the pilot in differentiating various auditory input and using redundant audio cues to reinforce proper interpretation of warnings.

c. Localization

The third type of audio imagery that can be presented is "Location of auditory cues in relation to exocentric objects". By presenting audio cues in relation to an object's location in three dimensional space, a pilot can instantly determine not only that there is a threat, but also the relative bearing of the threat [Ref. 6].

B. MERCATOR PROJECT

At the Georgia Institute of Technology, the Mercator Project seeks to give computer users who are visually-impaired access to software using a graphical user interface under the X-windows system. The idea is to map icons to auditory and tactile space [Ref. 12]. The system utilizes a set of HRTF's provided by Professor Fredric Wightman of the University

of Wisconsin at Madison. The interface is generally quiet, providing auditory cues only when requested or to inform the user of a change in state. Text files are presented to the user via a synthesized voice from a Digital Equipment DECtalk DTC01. This voice output is digitized and presented to the user in a spatial format. As a user navigates through the audio desktop, various background sounds are added (e.g. a fan, running water), allowing the user to navigate various spaces and maintain a sense of locality.

C. BACK TO THE FUTURE

Back to the Future is a motion-simulator ride at Universal Studios in Los Angeles, California [Ref. 13]. The system consists of two domes thirteen stories tall, with a 10,000 watt sound system in each dome. In addition, there are large clusters of speakers located behind an Omnimax screen, as well as speakers located inside the cars people ride. The cars are mounted on a hydraulic platform similar to aircraft flight simulators. The system uses pre-programmed MIDI data and sound effect samplers to generate spatial audio. It does not spatialize audio in real-time. The sense of locality experienced is based on pre-determined paths for the ride. One of the interesting developments is the use of "frequency injection," which sends very low-frequency sound waves in the 4 Hz range into the motion simulator, which greatly enhances the effect and sense of immersion. Similar applications have been installed at theme parks throughout the United States and are becoming more popular everyday.

D. VANDERBILT

At the Vanderbilt University Computer Center, Brain Evans has developed a method of enhancing scientific animations using sonic maps [Ref. 14]. The idea is to represent the animation with a musical score that represents the visual information presented. The data from the visualization is translated into rhythm and pitch data. The translated data is used to activate sounds from a synthesizer using the MIDI protocol. His interpretation of fractals generating a MIDI output in various time formats is probably the most well know presentation of this work.

IV. NPSNET-PAS

A. DESCRIPTION

The Naval Postgraduate School Networked Vehicle Simulator IV (NPSNET-IV) is the newest incarnation of a three-dimensional visual simulator developed at the Computer Science Department's Graphics and Video Laboratory. The project centers on the development of graphics simulation software and has expanded to include many facets of virtual reality [Ref. 15].

The NPSNET-PAS (Naval Postgraduate School Networked Polyphonic Audio Spatializer) is the program that serves as a link between NPSNET-IV and the sound system hardware. It is a combination of "off-the-shelf" hardware and student written software that is capable of generating audio cues with an approximate spatial location for NPSNET-IV. These audio cues are based on geographic and event driven scenarios that correspond to the actions of the players, autonomous forces, and the environment. The audio cues are presented to the user in the form of free-field (external loudspeakers) in a surround-sound configuration.

B. DEVELOPEMENT OBSTACLES

In general, there are two ways to deliver audio cues to a listener: headphone reproduction and free-field (loudspeaker) reproduction. In reviewing the associated problems with these types of systems, it is important to understand that these deficiencies are also present in the way humans determine spatial locations in the real world.

1. Headphone Reproduction

a. Problems

In order to deliver spatial audio cues via headphones, it is necessary to process enormous amounts of digital audio data. Since we only have two speakers the sound must be filtered using a HRTF. Commercially available hardware components to meet the computational requirements do exist. The most popular of these is Crystal River

Engineering's Convolvotron. However, even this device has limitations. With an aggregate computational speed of more than 300 million multiply-accumulates per second the Convolvotron is an impressive machine. However, even with this immense computational ability, it can only process four individual sound cues simultaneously. In a networked virtual environment with multiple entities this is insufficient. Other problems associated with this type of system include the fact that the HRTF filters created using the binaural recording method are specific to the individual and these filters may differ significantly from person to person. The use of different types of headphones may significantly degrade effectiveness [Ref. 16]. In addition, users suffer tremendously from front-back confusion (confusing a 000^0 sound source for a 180^0 sound source) [Ref. 16].

Another device recently made available on the commercial market is the Roland SDE-330 Dimensional Space Delay [Ref. 17]. This machine is constructed similar to typical music effects processors. Recent reviews are fairly favorable in rating it's ability to generate spatial audio. However, the device is capable of only localizing one input source.

b. Recommendations

One option would be to purchase several Convolvotrons or Dimensional Space Delays. This would give the capability to deliver spatial audio cues for multiple voices. However at a cost of $20,000 per unit for the Convolvotron and $1,000 per unit for the Space Sound processor, this can rapidly become cost prohibitive. In addition, problems inherent to spatial audio previously discussed exist even in these machines. Some of these, e.g. front-back reversal, can be overcome with the use of interactive head-tracking and taking advantage of the "ventriloquism effect" that results from seeing an event such as an explosion occur with an associated audio cues in an approximate spatial location.

2. Free-Field Reproduction

a. Problems

Presenting spatial audio cues via external speakers is currently being done in many formats. Dolby surround sound is perhaps one of the most common. This can be heard in movie theaters around the world.

Problems in generating these audio cues involve the selection of speakers and method of equalization. Mismatched speakers will severely degrade any type of spatial audio effect. Environmental reverberation can alter listeners comprehension of spatial locality and crosstalk can occur when both ears receive the same sound from both loudspeakers [Ref. 16].

b. Recommendations

Overcoming these inherent problems with Free-Field audio systems can be done by choosing quality loudspeakers that are as flat in magnitude as possible and nearly linear in phase. A properly designed room can prevent sound reflection and when combined with speakers placed in the most advantages position will increase the spread of the “sweet spot”, or effective listening area. [Ref. 16]

C. GENERAL OVERVIEW

1. Design Basis

The design for NPSNET-PAS was based on several factors. Regardless of how the audio cues are spatialized, the requirement for any system is some form of audio hardware to generate the sounds. This can be a computer capable of playing audio files, tape cassette player, CD player, or more commonly a digital sampler. A sampler is a device often with an attached keyboard that is capable of storing large quantities of digital audio and playing back the samples. The advantage to using a sampler is that they are constructed with the ability to respond to various control commands using the MIDI protocol.

The other common requirement across all systems is that each sound to be generated must have its own track or audio signal path. If the sounds are mixed at any point prior to delivery, any filter applied to that track is applied to all sounds in that mix.

Other factors affecting the choice between headphones and external speakers are that if headphones are used the audio generated serves only one user. With a free-field system, the cues are still spatialized based on one listener, but other users in the room can enjoy some of the benefits of the system. Additionally, to implement a headphone delivery system would have required the purchase of extremely expensive hardware e.g. Convolvotron, Dimensional Space Delay.

Since the lab was already equipped with a sampler for generating audio cues and external audio hardware with various mixing and routing capabilities, a system to take advantage of this hardware was designed. The use of the existing equipment significantly reduced the cost of the system. However, this was not the only factor affecting the decision. As previously mentioned, the commercially available products have limitations regarding the number of voices that can be localized. Analysis of the sampler available in the lab showed that it could produce sixteen simultaneous voices and that each of the voices could be given individual parameters for localization. Other features included ease of implementing control features available in the MIDI protocol, and the ability to continually increase the complexity and capabilities of the system with minimal cost.

The trade off is the resolution of the localization. In the commercial products, the resolution or spatial quality of the sounds is much more accurate than can be achieved with a MIDI sampler. For purposes of this implementation, a decision was made to have the ability to localize a greater number of voices with a lower resolution.

2. Overview of Data and Audio Signals

To better understand the system, let's first look at the data flow. In attaching audio cues to a virtual environment, the first objective is to extract the necessary information from the virtual world, e.g. geographic location, type of event. This occurs thru the use of the

DIS (Distributed Interactive Simulation) protocol [Ref. 18], which allows us to receive data packets over the Ethernet that contain the geographic and event data for all entities in the virtual world. The next step is to process the data. This data implementation utilities the

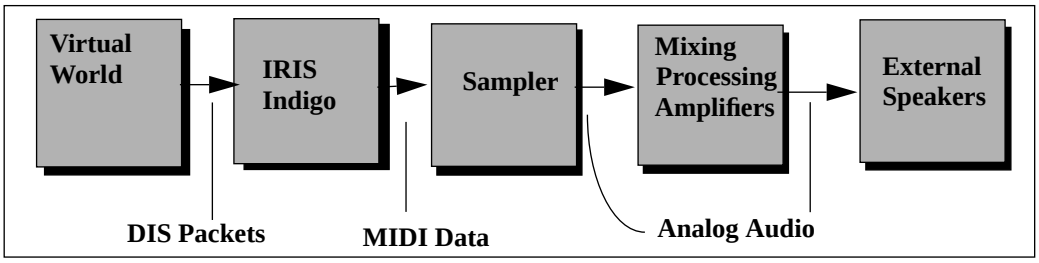


Figure 2: Data and audio overview for NPSNET-PAS

Musical Instrument Digital Interface (MIDI) protocol, which consists of building messages which are then sent to a sampler. Although initially designed for use in musical composition, this protocol serves to provide a fair amount of control over parameters necessary to spatialize audio. Based on commands received via MIDI, the sampler generates analog audio signals. These signals are then processed by digital effect processors, mixing equipment, and then routed to the amplification system which in turn is output via external speakers (see Figure 2).

The generalized data flow diagram is just the beginning of understanding how the system works. The sampler has the capability of generating audio on any one of 8 sub-channels. This allows for the creation of individual audio control channels that can be routed through external audio speakers strategically placed around the user. The following chapter discuss in greater detail the software and hardware used to take advantage of these capabilities.

V. SOFTWARE DESIGN AND FUNCTIONALITY

The NPSNET-PAS reads DIS packets [Ref. 18], then it processes the information from these packets to determine what type of sound events have occurred and generate audio from any of six speakers surrounding the user (see Figure 3). The data packets used in the DIS standard are called PDUs, protocol data units. The PDU contains the entity identification (host address, entity number, and domain), position, velocity, orientation, and appearance. This information can be used to determine the type of event and the location of the event. With a library that reads DIS packets in place, NPSNET-IV and NPSNET-PAS are able to handle information from any DIS-compliant simulator source. The library used for the NPSNET-PAS system was taken from the NPSNET-IV vehicle simulator [Ref. 15]. The DIS specification calls for twenty six different PDU'S. However, currently NPSNET-IV uses only the Entity State PDU, Fire PDU, and Detonation PDU.

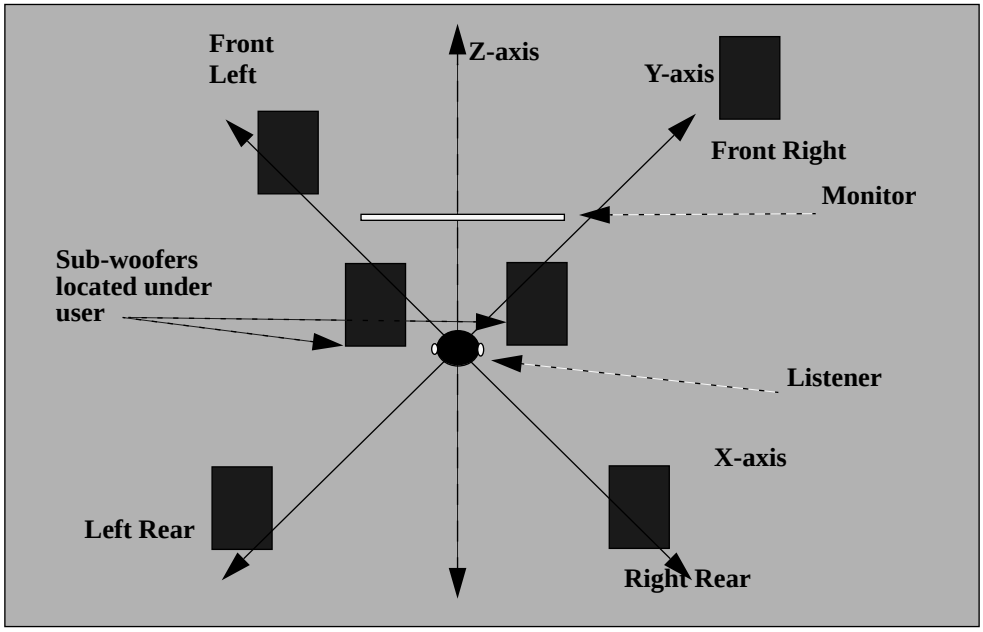


Figure 3: Speaker Placement for NPSNET-PAS

Appendix A Users Guide contains specific details for starting the program and fully explains the use of the various data files necessary for proper execution. In appendix B is

a more detailed description of the various MIDI messages generated by the functions described in this chapter.

A. MAIN FUNCTION

Once the program has been executed the function ‘main” of NPSNET-PAS in the file sound_main.C begins with several initialization functions. These include:

• *process_state*: Procedure to read the command line arguments chosen by the user.

• *setupwin*: If the user has chosen to display the output of NPSNET-PAS this function sets up the graphic window for display.

• *get_config*: Procedure to read the configuration file and determine exercise number.

• *get_host_data*: Procedure to determine host. All sounds generated will be based on the position of the selected host.

• *read_environmentals*: Procedure to read the environmental data file. This provides the geographic location of various objects that activate a sound cue when the user is within a specified range.

• *net_open*: Procedure to open network port. This can be done in either broadcast or multicast mode.

• *initialize_port*: Procedure used to open the MIDI port to allow communication to the sampler.

When initialization is completed, the program enters into the main loop where it begins responding to DIS packets and generating spatialized and non-spatialized aural cues based on the selected host. If the main loop detects at least one PDU, it begins processing the PDU. If there are no PDU’s detected, the program moves forward and processes the state of the program (see Figure 4).

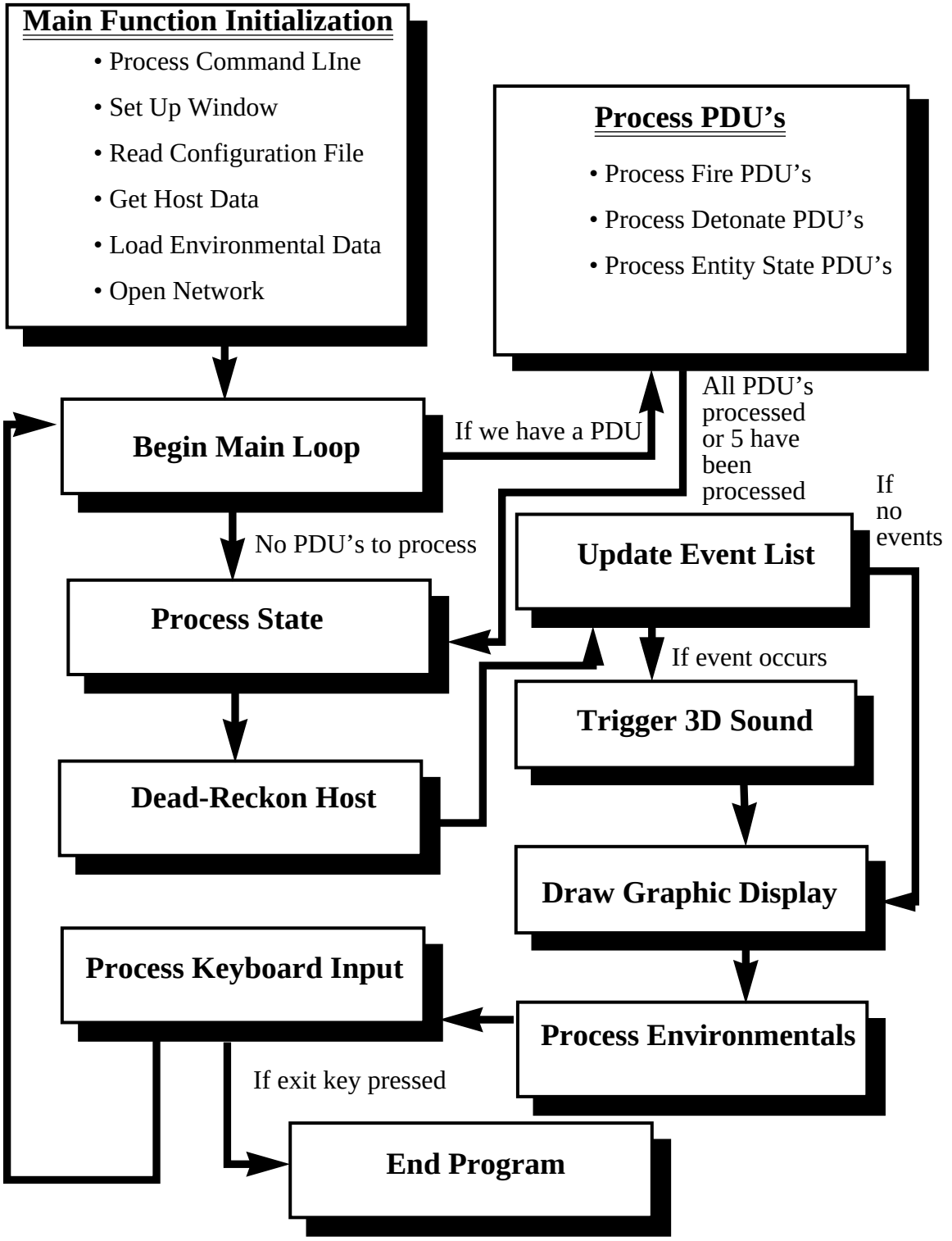


Figure 4: NPSNET-PAS Program Data

B. PROCESS PDU'S

When at least one PDU is received, the program enters into another loop to process the data. This processing loop continues until either all PDU's received have been processed or five PDU's are processed. PDU's have been known to arrive at the rate of 100 per second. Without some limit on processing per cycle, the program could remain in this loop indefinitely. The process PDU loop is set up as a case statement that looks for one of three cases. These are that we have an Entity state PDU, Fire PDU, or Detonation PDU.

1. Entity State PDU

When an entity state PDU is received, the *process_entitiyPDU* function is called. This function parses all the necessary information from the PDU into a record structure that NPSNET-PAS can use for processing. The information contained in the PDU can cause two different MIDI messages to be generated.

a. Vehicle Sound Actuation

After the loop to read PDU's is completed, a function called *process_state* is called. The function reads data in the entity record and determines if the host vehicle sound has been started. If the sound has not been previously started, it calls the function *my_sound_on* located in soundlib.C. This triggers the sampler to play the appropriate vehicle sound. There is additional functionality in *process_state* that is explained in greater detail in section C.

b. Vehicle Acceleration

The other action upon receiving an entity state PDU is to call the function *process_vehicle_sound*. The procedure determines if the entity PDU is in fact the host and, if so, then builds a MIDI message to alter the pitch of the vehicle sound based on the vehicle's current speed. This gives the user the impression of acceleration and deceleration. The algorithm to calculate the rate of pitch shift is as follows:

```
pitch = (int)((user.speed/(1.0*user.maxspeed)) * (double)(Max. Pitch);
  if (pitch < 0)
    pitch = 0;
  else if (pitch > Max. Pitch)
    pitch = Max. Pitch;
send(pitch);
```

A MIDI pitch bend message consists of three bytes. The first byte is the channel number to apply the pitch bend. The least significant 14 bits of the second and third bytes make up the amount of pitch bend to apply to the channel given in the first byte, the most significant two bits are thrown away.

2. Fire PDU

While in the process PDU loop, if a fire PDU is received, the program calls the function *process_firePDU*. In this function, the identity of the fire event, i.e. weapons type, is determined. The next check is to determine if the weapon was fired by the host. If it is the host firing, we want to immediately generate the sound effect, so the function *trigger_3D_sound* is executed. This function is discussed in greater detail in section F. If the fire PDU is from another entity, then the function called is *add_to_event_list*. This function puts the fire event into a list of records for all the fire events that have occurred. This event list contains information regarding the type of weapon fired, location of the firing, and the time the firing occurred. The event queue is processed in another step and is discussed in section E.

3. Detonation PDU

Detonation PDU's are handled in a similar manner to fire PDU's. The function *process_detonationPDU* is called. This function extracts the type of detonation, location of detonation, and time the detonation occurred. This information is also stored on the event list. Note the only difference here is that all detonation events are added to the event list, whereas only non-host fire events were added. All this means is that all detonations will have a time of arrival calculation and the host firing events will be processed immediately.

C. PROCESS STATE

Once the current list of PDU's has been processed, the program executes the function *process_state*. This function performs several control features. Previously it was stated that in this function the program does a check to determine if the host vehicle sound has been turned on. That is one of the control features in this function. Actually there are several modes the program can enter based on the current status of entities.

• loading - In this state, NPSNET-PAS has just been executed. The program is loading the sound bank. Once this is complete, a MIDI message triggers a sequence to play a voice sample informing the user of system activation.

• no_vehicle - This state is used initially if the system has been activated and no vehicles have been detected. This will generate a MIDI message to load and play a sequence of music. The music will continue to loop until another mode change occurs.

• alt_vehicle_alive - While in the no_vehicle state, if the system receives a packet from an alternate vehicle, it will use that entity as a host and turn the appropriate vehicle sound on.

• my_vehicle_alive - Once the host vehicle is detected, the program will switch to this state and turn the appropriate vehicle sound on.

• alt_vehicle_dead - If the system has been in the alt_vehicle_alive mode and it receives an entity PDU informing the system that the vehicle was killed, the program will switch to this mode and play a serious of explosions.

• my_vehicle_dead - If the system has been in the my_vehicle_alive mode and it receives an entity PDU informing the system that the vehicle has been killed, the system will switch to this mode and play a series of explosions.

Once in a particular mode the program will stay there until there is a mode change. This prevents the system from continuously turning on a particular sound. This can have a drastic effect on the sampler and is to be avoided at all cost.

D. DEAD RECKON HOST

After the *process_state* function is complete, the dead reckoning algorithm is used to update the position of the user. This allows for much more accurate calculation of intersections between the user and the various sound events. An entity under the DIS standard only sends out position information if there is a change in course, speed, or it has not sent one in five seconds. If a dead reckoning algorithm were not used, net traffic would have to be significantly increased in order to maintain accurate position data. The algorithm is a basic implementation of adding the respective x, y, and z velocity times the change in time since the last update x, y, and z positions to the current x, y, and z position.

E. UPDATE EVENT LIST

When a fire or detonate PDU is received, NPSNET-PAS loads the sound event onto a list of events to be processed. After the vehicle position has been updated, the function *update_event_list* is called. The function consists of a loop that screens each event on the list, calculating the radius of the sound event by multiplying time since it occurred by the speed of sound.

```
radius = ((current_time - previous_time) * SPEED_OF_SOUND);          Eq 1
```

SPEED_OF_SOUND was defined for sea level at 70 degrees Fahrenheit, in air, 335.28 meters per second. The radius of the sound wave-front is compared to the distance from the user and one of three tasks is carried out. If the sound is determined to be out of range of the user, it is deleted from the list. A distance of 12700 meters is used for this parameter. This makes the assumption that if the sound event is beyond this range, the user will never hear it. If the sound has intersected the user's position, the sound event is passed to the function *trigger_3D_sound*. This will generate the MIDI message that will play the sound effect corresponding to the bearing and range of that particular event. The third option is that a sound is within range to be heard by the user but the sound wave has not arrived. In this case the sound will remain on the list.

F. TRIGGERING 3D SOUND

The trigger_3D_sound function processes the position of the sound, the position of the listener, and the note number of the sound to be played. It then turns this information into the correct series of MIDI note-on and note-off commands based on the distance of the sound from the user and the relative bearing to the user. First, it computes the distance between the sound source and the user. The next step is to determine the intensity or loudness of the sound to be played.

1. Sound Intensity

The determination of sound intensity is difficult. The volume scale used was derived from the work of Durand Begault [Ref. 19]. In his work, Begault conducted several experiments in half-distance perception. The basic idea was to play a tone at some dB, then reduce or increase the dB and ask the listener whether the perceived change in volume resulted in the perception that the sound had moved twice as far away or 1/2 the distance closer. The result of Begualts' work indicates that a reduction of 6 dB rather than 3 dB (physically based inverse square law) resulted in much improved perception of half-distance. This resulted in the following formula for volume:

Volume =1 -(((logMax_Range/Half_Dist) / (logMax_Range/Half_Dist)) * total_volume) Eq 2

Max_Range comes from the maximum range at which a sound can be heard. Half_Dist is a constant used to represent the distance in which loudness decreases by 6 dB. Total_volume is the maximum Midi velocity (volume) number, which ranges from 0 to 127. This formula calculates the number of half-distances away the listener is from the source, then normalizes this number by the total number of half-distances within the Max_Range, using the Half_distance number as the first half distance. The normalized number is now subtracted from 1 to give the appropriate percent volume that should be multiplied by the total_volume. In essence, the logarithmic nature of the loudness is converted to a linear scale for use with the linear MIDI volume range. Judicious choice of Max_Range and Half_distance will control the rate of drop-off. For now, Max_Range has

been hard coded to be 12,700 meters and Half_Distance is set at 50 meters. These numbers were chosen in an attempt to correctly reflect sounds experienced in the virtual world. Since the current system is based on war simulations, the majority of the sound effects are loud explosions and cannon blasts that have the characteristic of being heard at long distances. In reality, these factors would be different for every sound in the sound effects bank.

2. Directional Calculation

Once *trigger_3D_sound* has computed the total volume of the sound to be played, it performs the calculations necessary to determine the relative bearing to place the sound. Since we are using speakers that surround the listener, the idea is to calculate and generate MIDI messages that will play the sound in the necessary speakers at the determined value.

a. Correction for speaker offset

The external speakers surrounding the listener are offset by 45 degrees. Positive X-axis is the forward right speaker and negative x-axis is the left rear speaker. The y-axis run from the forward left speaker to the right rear speaker. The x and y coordinates are altered by 45 degrees to compensate for this correction. The algorithm for this is as follows:

```
temp1 = x;
temp2 = y;
x = (temp1/SQRT2) + (temp2/SQRT2);
y = (temp2/SQRT2) - (temp1/SQRT2);
```

b. Correction for vehicle orientation

The next step is to correct for the vehicle's orientation so that the coordinates are in terms of the user's position. This is the final step to correcting the x,y, and z coordinates of the sound source. Without this correction, the sound source would not be set to the proper angle of orientation and the sounds would be placed incorrectly. The variable my_view is an array of the users postition and orientation read into the function *trigger_3D_sound.* The corrections are done for all three angles psi, theta, and phi:

```
/* correct for orientation of vehicle */
/* rotate by psi */
temp1 = x;
temp2 = y;
x = (temp1 * cos(my_view.psi)) + (temp2 * sin(my_view.psi));
y = (temp2 * cos(my_view.psi)) - (temp1 * sin(my_view.psi));

/* rotate by theta */
temp1 = x;
temp2 = z;
x = (temp1 * cos(my_view.theta)) - (temp2 * sin(my_view.theta));
z = (temp2 * cos(my_view.theta)) + (temp1 * sin(my_view.theta));

/* rotate by phi */
temp1 = y;
temp2 = z;
y = (temp1 * cos(my_view.phi)) + (temp2 * sin(my_view.phi));
z = (temp2 * cos(my_view.phi)) - (temp1 * sin(my_view.phi));
```

c. Amplitude Variance

The total volume calculated using the volume algorithm must be divided up among the speakers. The function calculates the percentage of the total volume that must be played in each speaker to place the sound source correctly by multiplying the total volume previously calculated by the angle between each respective axis x,y, and z and the vector from the origin of the user to the sound source, theta:

$$A_i = A_{total\_volume} \, COS(\alpha) \qquad \text{Eq 3}$$

This is the same as the ratio between the length of the particular component axis to the length of the distance vector to the user. The following function is calculated for each of the three axis:

$$A_i = A_{total\_volume}(\text{x,y, or z length/distance to source}) \qquad \text{Eq 4}$$

This gives us the percentage of the total volume to be played for that respective axis. This percentage is then:

$$\text{volume } A_i = \text{floor (total\_volume} + (20\log(A_i / \text{total\_distance})) \qquad \text{Eq 5}$$

The number twenty is an agreed upon sound pressure reference for determining sound intensity at the eardrum [Ref. 20].

d. MIDI Message construction

The last step for *trigger_3D_sound* is to construct three note-on and three note-off MIDI messages, one note-on message and one note-off message for each axis x, y, and z. Each of the six messages consists of three bytes. The first byte is the channel

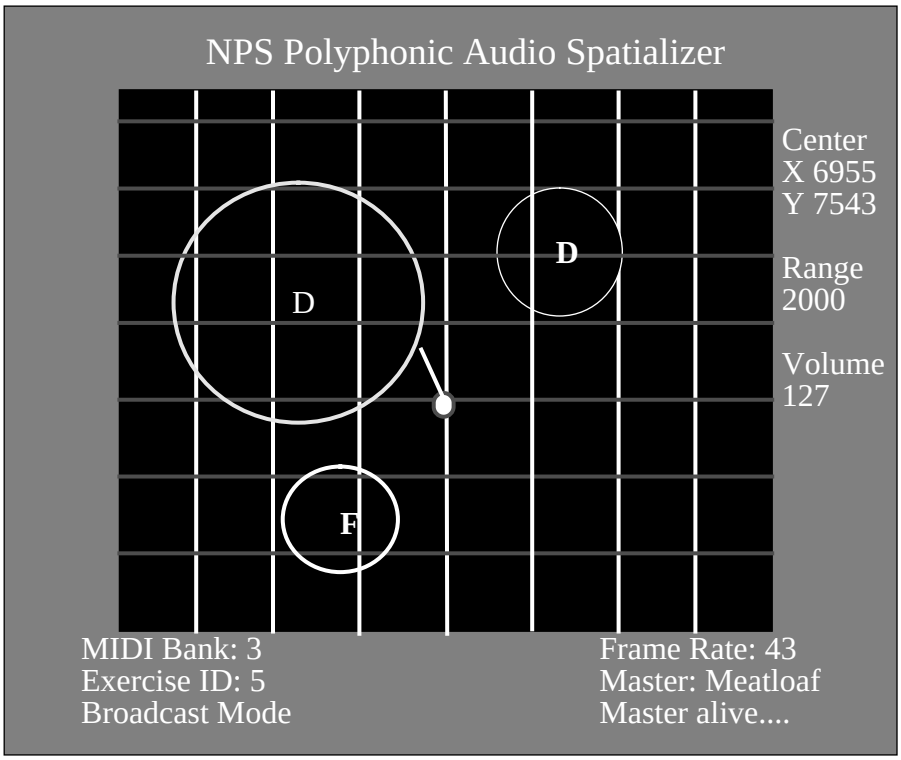


Figure 5: Graphic Display of NPSNET-PAS

number and status, i.e. note-on or note-off, which also corresponds to the speaker to be played. The second byte is the note value, e.g. explosion, cannon fire etc., which was passed into the function. The third byte is the calculated volume for that particular axis, or zero in the case of a note-off command. Once the MIDI message has been built, the message is transmitted to the sampler.

G. GRAPHIC DISPLAY

When NPSNET-PAS is started, the program opens a two dimensional graphics display (see Figure 5). The initial screen is an introduction to the program and informs the

viewer of the sound bank being loaded, assigned host, and exercise number. Once the program initializes the MIDI port and loads the sound bank, the display is switched to a two dimensional grid representing the terrain of the virtual world from a god's eye view of the xy plane.

The display draws an icon representing the host. This icon has a short line protruding out that indicates course or the direction of travel. The initial value of the grid is one thousand meters from center to edge. This parameter can be increased or decreased. The position of the user is static, in that the grid moves and leaves the host at the center of the view at all times.

When a fire or detonation PDU is received by the program, an "F" for fire or "D" for detonate is drawn at the position of the event. As the sound wave is tracked and monitored for intersection by the function *update_event_list*, a circle is drawn that emanates from the initial position. This circle's radius increases until the sound event intersects the host.

The display also displays the x and y coordinates of the host, the range of the geographic grid, the maximum volume variable, frame rate, master machine assigned, status of the master, MIDI bank loaded, exercise ID, and network mode.

H. PROCESS ENVIRONMENTALS

After the program returns from the *update_event_list* function, the next step is to process PDUs for environmental effects. Environmental effects consist of sounds played to reinforce a user's location in relation to the terrain or specific geographic locations such as a lake, farm house, forest or any other significant locality. These events can take the form of sound effects generated by the sampler or a change in the acoustics of the sound system in general.

1. Environmental Sound Cues

The function *process_environmentals* uses an environmental file that has been read and stored in a list during program initialization. Each environmental effect on this list consists of a note number for the assigned sound, a geographic location of the

environmental effect, and a radius about which the effect will be heard. A simple calculation compares the user's location to the location of the sound's source. If this distance is less than the assigned radius, the request information is passed into the *trigger_3D_sound* function and a spatial sound cue is played reflecting the environment of the geographic location, e.g. crickets chirping, waterfall. A safety feature in the form of a timing mechanism has been added to prevent the program from repeatedly regenerating the sound. This safety feature insures that a MIDI play message is only sent every four seconds while the user is inside the radius of the sound source. Without this, the program would continuously generate note-on commands over and over, thus seriously overloading the sampler.

2. Environmental Acoustic Effects

A recent addition to the sound system is two digital signal processors. These are capable of adding various acoustic effects, e.g. reverb, delay, phase, chorus, flange. A simple bounding box is created using two sets of x, y, and z coordinates. If the user enters into the bounding box, a MIDI message is constructed and sent to the signal processors that causes a program change. There are ninety-nine different configurations that can be set for various levels of effects processing. Each of these has the effect of texturing the basic sound, e.g. large reverberation for canyons and caves, flat texturing for small rooms with no reverberation.

I. PROCESS KEYBOARD

The last function called in the main loop is *process_keyboard*. This function takes input from the keyboard and executes preassigned control functions. The control features consist of the following:

• Override Mode - pressing the "o" key will put the audio system in override mode. Sound cues will cease and the sampler will play a looped musical track. Pressing the "o" key again will return the system to normal.

• Master Volume Control - The up arrow and down arrow keys will raise or lower the maximum volume parameter used to calculate a sound cue,s volume. This parameter is initialized at 127, which is the maximum for MIDI.

• Grid Size - The pad plus and pad minus keys will alter the scale of the geographic view grid.

• Exit Program - The escape key will exit the program.

VI. HARDWARE DESIGN AND FUNCTIONALITY

A. GENERAL DESCRIPTION

All of the hardware being used consists of readily available "off-the-shelf" audio components. The current configuration consists of an Iris Indigo Elan, an EMAX II sampler, one RAMSA 6-channel mixing console, one RAMSA sub-woofer pre-amp, two RAMSA Power Amps, one Carver Power Amp, two RAMSA 2-way Speakers, two Infinity Studio Monitors (audio speakers) and two RAMSA sub-woofers (see figure 6). In order to effectively understand how to set up and wire the system, Appendix C contains a series of drawings with a much greater level of detail.

B. EXTERNAL AUDIO COMPONENTS

1. Computer to Sampler

Once the software has generated the required MIDI commands, they are sent out via an RS-422 port on the back of an Iris Indigo. The Elan has two ports designated as ttyd 1 and 2. These ports are capable of RS-422 communications and can be used for transmitting and receiving MIDI data. The next stop is an Apple MIDI converter that converts the signal from 8-pin RS-422 to RS-232 protocol and 5-pin MIDI cable output. This cable is then attached to the "MIDI in" of the EMAX II sampler. The EMAX II is capable of receiving MIDI data on 16 different channels and generating 16 notes at a time. This is often referred to as being multi-timbral.

2. Sampler Set Up

The sounds on the EMAX II are organized into banks, which are further subdivided into presets. It is the presets which serve as a basis for our sound generation. To play sounds on the EMAX II, the user loads a bank. A bank consists of anywhere from 1 to 99 presets. Once a bank is loaded into memory, any one of the presets stored in that particular bank can be accessed. This allows the user to play the keyboard with that particular sound effect, e.g. piano, flute, cannon blast.

To allow the EMAX II to receive on multiple channels and respond by playing multiple instruments or, in our case, multiple sound effects, we have to build a sequence. Essentially the user puts the sampler in the sequencing mode, loads a preset, for example trumpet, then records about 1 or 2 seconds of blank sequencing time on the desired channel. This is done for each successive track or channel that will receive MIDI data.

Once the sequence is built, we need to do some manipulation of the presets. For each presets used, one of the four available sub-output channels (A, B, C, Main) is selected. The output of these various presets can be routed to any one of the four sets of output channels. Once the submix has been selected, the preset definition settings are used to remove the stereo effect of the voices and then pan them left or right depending on the requirement.

To those who have never worked with a sampler, this may seem a bit confusing. Perhaps the following example will help to clarify the end result. If we wish to generate a missile firing sound effect, the NPSNET-PAS software computes the necessary values and generates a MIDI message. Our MIDI message consists of anywhere from 1 to 3 note-on commands. The messages contain the necessary information for the sampler to respond to individual channels and corresponds to the settings we have made in the presets. If the missile firing had occurred forward and to our left, the message generated would have given us a note-on command to the note and channel number of the missile firing sample set up to generate sound for the forward left speaker. We would hear the sound forward and to our left. Appendix D contains additional information of the sampler set up and detailed instruction for manipulating the sequence, voice, and preset settings.

3. Sampler to Signal Processors

Once the sounds have been generated, the submix outputs are passed to two digital signal processors. The signal processors contain four Digital Signal Processing (DSP) chips each and are capable of generating acoustic effects and equalization on four individual input/output channels. Since there are two processors, we can control eight separate channels. This capability is necessary because most signal processors take multiple

instruments and mix them down to two stereo outputs. This would eliminate our spatial effects by combining all of the inputs into two outputs. The signal processors also respond

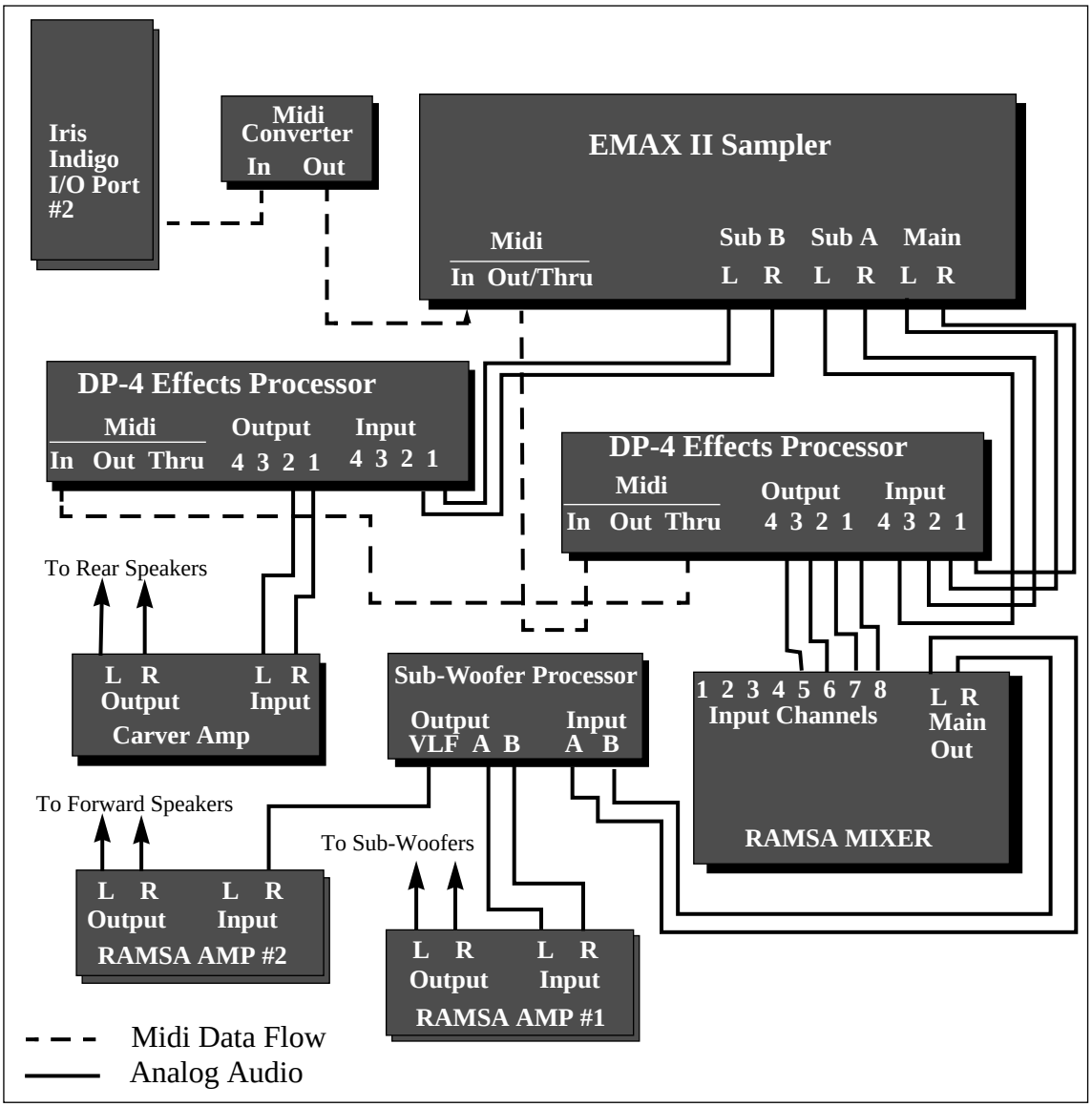


Figure 6: Hardware Design

to standard MIDI program changes. This permits us to change effects algorithms in real time. From the output of the DSPs, the audio signals are routed to either the mixing console or the external amplifiers.

The other hardware routing from the sampler to the signal processors is a MIDI control line. This is connected to the output of the sampler and then routed to the input of effects processor number 1. From the MIDI output of processor number 1 another MIDI cable is routed to the MIDI input of effects processor number 2.

4. Signal Processing to Amplification and Mixing

Output signals from the effects processors take one of two paths. Submix output Main and Sub-channel A are routed into the mixing console. Submix channel B is routed to it's respective external amplifier.

It is important to understand that the audio signal for each channel left and right must maintain its respective signal path without being mixed into other channels. Each amplifier used merely passes the audio signal through left and right channels, no mixing occurs.

The only exception to this is the vehicle sounds coming from the Main channel. These signals along with those of sub channel A are mixed together in the mixing console. Even though these channels are mixed the left and right signals are kept separated. This was done to project some of the vehicle sound thru the forward speakers. When only played through the subwoofers, the audio signal has a tremendous low frequency rumble, however it lacks the necessary clarity to allow the listener to determine the type of vehicle being driven.

From the main output of the mixing console the signals are routed to the sub-woofer processor. This acts as a very low-frequency (VLF) filter or crossover. The audio signals are filtered and one output is the VLF. This signal is routed to one of the RAMSA amplifiers and then onto the sub-woofers. This amplifier is set up in an AB mono bridge mode that allows the single channel signal to be routed to both the left and right sub-woofers. Two other outputs of the sub-woofer processor are the A and B. This is a mimic of the original left and right audio signal, which is routed to the other RAMSA amplifier and then to the forward left and right speakers.

VII. IMPLEMENTATION ANALYSIS

Although there were quite a few calculation and timing issues, the program manages an average of 30 to 50 cycles per second through the main loop, resulting in no perceived degradation of real time response. From a user's standpoint, the system has been effective in achieving sound localization using amplitude modulation. In addition, the delay of arrival time based on the speed of sound has added additional realism to distance perception.

A. SPEAKER PLACEMENT

The placement of external audio speakers is the subject of much debate. The first model constructed was a sound tetrahedron. The basic idea was to place three speakers on the horizontal plane in a triangular configuration, and a fourth speaker suspended over the head of the listeners. Several test runs indicated that this was not effective. The physically based model for this layout appears to work. However in practice, the wall reflections, early echoes, and baseline room noise totally destroyed any attempt to effectively localize sound cues.

A second design was implemented using a grid similar to the x, y, and z coordinates of a three dimensional graphics display. The x and y axes were placed front to back and left to right, respectively. The z axis was placed above and below the listener. This design met with some success and was implemented for a couple of weeks. However, the spatial effects were not as dramatic as expected. In addition, when adding this system to a vehicle simulator with a large screen display in front of the listener, the placement became impractical.

The third attempt at speaker layout was to take the previous design and rotate the x and y axis such that the forward and rear speakers were offset by 45 degrees from the listener. This configuration met with much success and the system has maintained this configuration. One significant problem that remained was the z axis speaker channel. As a general rule humans spatialize fairly well on the horizontal plane, however the vertical

plane can cause a lot of problems. Experience in the lab with speakers placed on the vertical axis have not shown this placement to be very effective. This part of the system has been temporarily removed and the focus of programming has been to localize strictly on the horizontal plane. One solution to the vertical problem would be to place the speakers in a cube like configuration, by placing one speaker in each corner of a square room. Currently lack of space and hardware have prevented this type of configuration from being tested.

The subwoofers were initially placed somewhere near the forward channel. This was somewhat effective. However, when they were moved to as near the base of the user as comfortably possible, the results of the low-frequency injection were remarkable. The earth shaking rumble from the sub-woofers has a hypnotizing effect on the users. Since one of the primary goals of the audio system is immersion, they have remained in this location.

Another factor in speaker placement for the forward and rear channels is the distance from the listener. The farther away from the listener, the greater the area affected, thereby increasing the "sweet spot", the listening position which gives optimum effect of spatial effects. The current room design allows a distance of 7 to 8 feet from listener to speaker.

B. SOUND INTENSITY

Determining the volume at which to play specific sounds based on range from the listener is a difficult calculation. Two different algorithms have been used to make this calculation. Each algorithm has an advantage and a disadvantage (see Figure 7).

1. Initial Algorithm

Since the loudness portion of the note-on command, range of 0 to 127, corresponds to a linear scale, the first method was to take the log of the intensity to give the velocity:

$$\text{velocity} = 20\log10((AB)/r) \qquad \text{Eq 6}$$

A and B are constants. A is determined by the fact that the largest possible value for velocity is 127. So, 127 = 20 log(A), or A = 229,000. B is equal to the value of r (in meters) that corresponds to the loudest volume. For instance, if B = 50, the velocity will be 127 or larger for any sound less than 50 meters away.

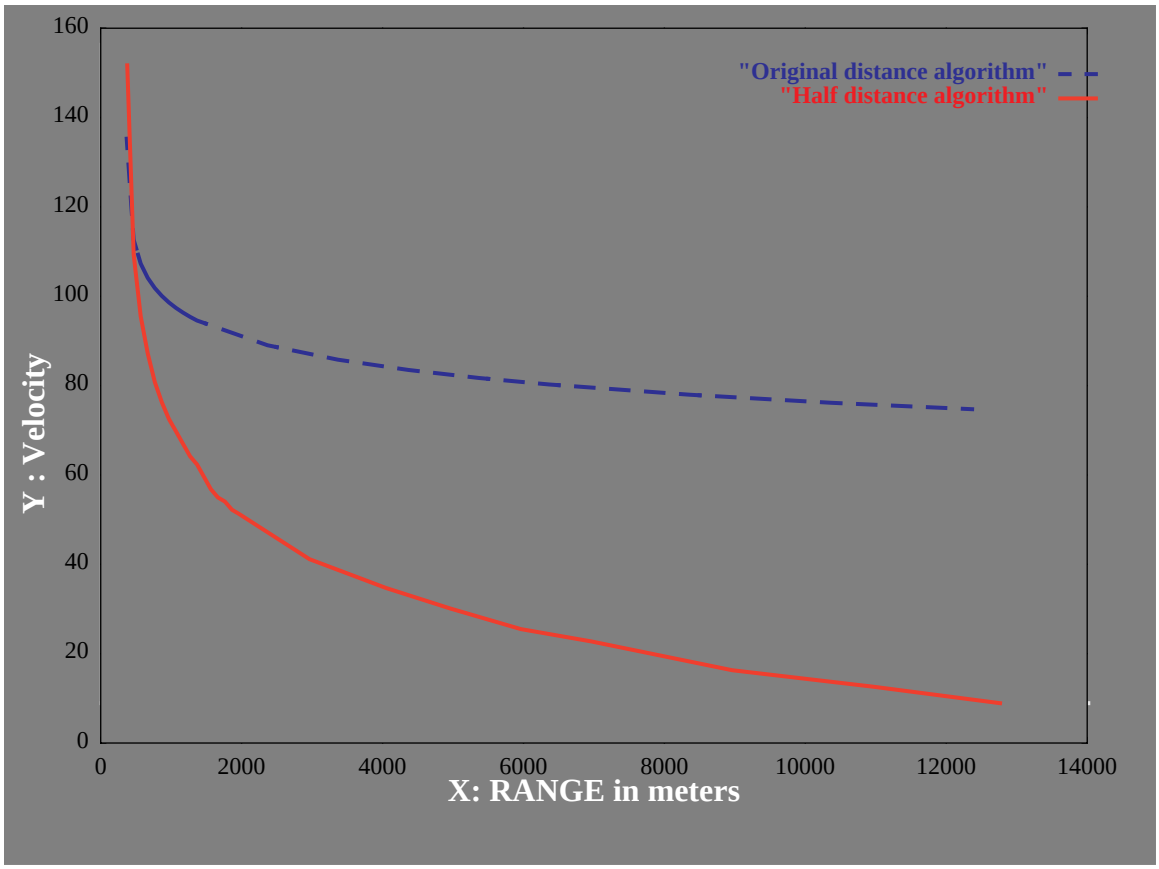


Figure 7: Volume Algorithm Comparison

This algorithm has worked, however there are some limitations. The rate of drop off does not occur as fast as predicted. Using a maximum range of 12,700 meters, sounds are still heard at a fairly significant volume even when at the outer limits of the maximum range. This has the advantage of playing a lot of sound cues and making the virtual experience for the user very entertaining. The disadvantage is that sounds are not placed correctly and the amplitude of the sound cue does not provide the user with any sense of realistic distance.

2. Second Algorithm

Discussions with the E-mu Corporation revealed that there was no data on the mapping of the analog audio output to the 0 to 127 MIDI scale. A volt meter was applied

to the analog output and measurements were taken of this output by sending a note-on command at intervals of 10 from 0 to 127. The results indicated that the analog audio output was near linear. The curve did become exponential at the very top and bottom of the scale. Since this scale was linear the task became to map this to an exponential scale similar to the inverse-square law. It is obvious that sound source volume decreases as the distance increases from the listener. Sound Intensity can be defined by the formula:

$$I = W / 4\pi r^2 \qquad \text{Eq 7}$$

Where I is sound intensity in watts per centimeter squared, W is the sound power in watts and r is the distance from the source in centimeters [Ref. 20].

Several experimental algorithms were tested and the best resulting formula was as follows:

Volume =1 - ((logMax_Range/Half_Dist) / (logMax_Range/Half_Dist) * total_volume) Eq 8

This formula was previously explained in chapter 5 section F. The algorithm is a much more accurate model of sound intensity drop off than the previously implemented algorithm (see Figure 7). The graph shows a dramatic difference between the initial algorithm and the one currently being used. The advantage of this algorithm is that a much more accurate model of sound intensity is constructed and the users are able to use the loudness to actually determine a sense of distance associated with a particular sound cue. The disadvantage of this algorithm is that the aesthetic value of numerous sound effects is lost. Since the algorithm more correctly reflects the physical behavior of sound, many sound cues that were previously heard at a considerable volume are now barely detectable or not heard at all. This is not necessarily a good or bad result. The issue now becomes whether you are trying to generate a physically correct model or an entertaining model. One solution for this is to install a software switch that would allow the user to switch to either algorithm depending on the type and purpose of the graphic simulation.

C. AUDIO HARDWARE

When we first constructed the system, we used a mix of various amplifiers and speakers which severely degraded system performance. Due to the dramatic difference in tonal quality of speakers, even the most casual listener could determine when sounds were spatially shifted. This has been improved with the addition of the RAMSA speaker system for the forward and sub-woofer channels and a pair of Infinity studio monitors for the rear channel. Future budgets allowing, the system will use matching speakers for all channels. Room construction and speaker placement have not afforded any way of eliminating cross talk between speakers nor of reducing the impact of wall reflections. What the system does do is give the ability to control when and where sounds are heard in relation to events occurring in a virtual environment.

Another difficulty incurred is the limitations of the sampler. Overall the EMAX II is a very capable sampler; however, it's original design was based on use in the music industry, not for generating sound effects in a virtual environment. Since only 16 voices can be generated simultaneously, when a 17th voice is added, the system begins dropping sounds. This can be extremely annoying. One would think that with 16 voices, the system would never become overloaded. However, just generating two missile firings forward, a waterfall behind the user, two explosions to the left and right of the user, and the user's vehicle sound can take eleven voices. Just a few more events and the system has reached maximum capacity.

One solution to this problem is to make the software more intelligent. Currently, when a note-on is generated, it is immediately followed by a note-off command. The sounds on the EMAX II are configured to play the entire sample based on this trigger action. If the program were more intelligent, it could fork off these processes individually and be smart enough to know the length of the sample it is playing and know when to generate the note-off command. This would give us much greater flexibility in managing the sound scene. A system of priorities could be developed that would prevent continuous sound, i.e. vehicles, from being bumped off. Also, we could avoid saturating the listener.

D. ACOUSTIC MEASUREMENTS

Measurements of spatial effectiveness and room acoustics were taken using a Hewlett Packard 3566-5 dynamic signal analyzer. A microphone was placed 2 feet in front of one of the speakers. We then played a cannon blast at a fixed range of 100 meters and moved the direction of the blast around the user in 15 degree increments. We captured the peak decibel readings for each sound event and compiled the data to produce a 3 dimensional graph. These graphs can be seen in Appendix E.

The experiment was for all four external speakers that make up the xy plane. The results indicate that the program does in fact spatialize the audio cues. One apparent result of this experiment was the determination that the roll-off for an individual speaker is fairly sharp. This is very obvious when observing the graphs; however, from the listeners' standpoint, when the other speakers are added into the equation vice analysis of a single speaker, this deficiency is less apparent.

One final acoustic experiment was to generate white noise and various other tones, including explosions and music, through all speakers at equal volume. The analyzer was used to project a real-time water fall plot which allowed the observation of frequency response deficiencies. The most notable was a drop in response centered around 55Hz. This corresponds to the crossover frequency of the speakers. In an effort to compensate for this deficiency, we equalize the sound using the parametric equalizer function provided by the digital signal processors. When a 55Hz centered frequency was boosted approximately 12 dB, the overall sound quality improved dramatically.

VIII. CONCLUSIONS AND RECOMMENDATIONS

A. FOLLOW-ON WORK

There remains a significant amount of work to improve the system.

1. Room Acoustics

Room acoustics could be vastly improved by the addition of sound deadening material to the walls. This would eliminate the effects of early echos and excess reverberation and allow for control over these factors using the digital signal processors. Currently the room being used is open to a larger portion of the lab. Installing a temporary wall to shape the room into a cube would also improve the external speaker response and eliminate external noise sources, i.e. large computer fans, air conditioning system.

2. Sampler Improvements

The EMAX II can be extended to increase the multi-timbral capabilities by adding any number of rack mount versions of the sampler. This is basically an EMAX II without a keyboard. These can be chained in conjunction with the existing EMAX II and increase the timbral capabilities at a rate of 16 voices per unit added. One feature of the EMAX II that has not been applied is the use of cross-fading. This can be applied in two different ways. One is to allow the sampler to pan a voice from left to right based on a MIDI command from 0 to 127. The second is to apply the cross-fade to a primary and secondary voice. Each sample on the EMAX II can contain two voices, a primary and a secondary. A particular sample such as a cannon blast could contain a primary voice that is a sample taken at a fairly close range. The secondary voice would contain a sample of the same cannon taken at a greater distance. The cross-fade value could be applied to the volume range of 0 to 127 so that commands of a given volume correlating to a close distance play the near sample and those of the volume correlating to a greater distance play the far sample. This cross-fade parameter can be set at any cross-over point between 0 and 127. Another improvement for the sample would be to create a data structure in the

trigger_3D_sound function that would intelligently track the number of voices being used and when the maximum is exceeded it would intelligently determine which sounds to stop and which to continue to allow to play. There are multiple ways to make this decision e.g. longest playing, greatest distance away.

One final improvement to the sampler would be to obtain a digital audio tape (DAT) machine and take samples in the field from the actual vehicles and weapons being emulated in the simulation. This would allow for knowing the distance and intensity of each sample recorded. This value could then be inserted into the current volume algorithm to improve the physical model for determining loudness.

3. Speaker Configuration

As mentioned earlier, one method of attempting to obtain vertical localization would be to configure the speakers in a cube fashion, placing one speaker in each corner of the room. This is fairly simple to implement in hardware, however a completely different algorithm for localization would have to be developed.

Another addition to the audio hardware would be a more capable mixing board. There are several relatively inexpensive mixing consoles that have the capability of creating subgroups from the inputs. One of the problems with the current set up is that the signal processors are wired in-line. When they shift effect algorithms there is about a half of a second where no sound is heard. With a more capable mixing board the effects processor could be wired as additive rather than as an in-line filter.

4. Indigo Audio

Currently there is a version of NPSNET-PAS that runs on an Iris Indigo that instead of sending MIDI commands to the sampler, merely triggers the play back of previously stored audio files. Although the sounds are not localized, this has allowed greatly increased portability.

Recently a public domain set of HRTF filters was released by M.I.T. and these filters can be used to generate spatial audio on the Indigo platform. The calculations to

apply these filters to various sound files in real-time are enormous. In order to implement this system in real-time, the basic requirement would be to take a given audio file and a set of HRTFs, combine these using software such as Math CAD, which is capable of applying a digital filter like a HRTF to a digital sound file, and generate an audio file of the sound for each of the possible localities. This involves floating point calculations in the hundreds of millions and would probably take a few weeks to generate for a half a dozen samples. Once this is complete, the audio files could be placed into a look-up table. When NPSNET-PAS receives the command to play an audio file, it could do a table look-up to determined the audio file with the closest match to the bearing and range of the sound event.

B. CONCLUSIONS

In implementing a physical model for three dimensional audio reproduction, there are many factors to be considered. The correct physical audio model does not necessarily result in the correct perception by the listener, due to factors present in the real world. Humans localize sound at varying degrees and some cannot localize at all [Ref. 3].

The loudness and time delay calculations are a good example. Since humans are susceptible to changes in loudness and time of arrival, these characteristics play an important role in the perception of distance. The perception of the sound intensity increases or decreases. Time of arrival produced in this system is sufficient for users to perceive a sense of distance. However many users commented that it detracted somewhat from the entertainment aspects of the system.

NPSNET-PAS does in fact generate spatial audio cues for a virtual environment. When the NPSNET-IV virtual environment is demonstrated with NPSNET-PAS audio system running, players experience increased levels of immersion, which leads to a greater level of interaction with the simulation. The system has it's limitations. However the design and construction process is ongoing and NPSNET-PAS can be expanded to overcome some of these limitations. Perhaps one of the greatest benefits of this research is that with NPSNET-PAS and a few inexpensive "off-the-shelf" audio components, the sensation of spatial audio can be added to any virtual environment utilizing the DIS network protocol.

The system is currently running at the Naval Postgraduate School Graphics and Video laboratory and at the Rand Corporation in conjunction with the JLINK program. Future installations have been discussed with the Army Research Laboratory and the Interactive Simulation Training Lab at the University of Central Florida.

APPENDIX A: USERS GUIDE

This appendix contains the necessary information to set up and run the NPSNET-PAS hardware and start the program. It is highly recommended that all users of the system read this information prior to using the system. Improper set up and execution can result in damage to the audio hardware.

A. HARDWARE SET-UP

The following items are required to be in the defined position or set-up configuration before starting the NPSNET-PAS:

- **Step 1 - SCSI Removable Hard Drive** - This is the SCSI hard drive that is attached to the EMAX II. This drive must be turned on before the EMAX II. The on/off switch is located in the upper right hand corner of the rear panel. When facing the front of the drive this would be on the left side. Once this drive is turned on the yellow lights on the front panel will begin blinking. When the drives have successfully booted the green lights will be lit and the yellow light extinguished. This operation will take approximately 20 seconds.
- **Step 2 - EMAX II Sampler** - Move the slider marked “VOLUME” to the lowest position possible. Facing the front of the EMAX II the on/off switch is located on the back panel to the right. Turn this switch on and allow approximately 25 seconds for the EMAX II to boot. Once booted press the button marked “SETUP”. The LED readout will show the words “Sequencer Setup” in the top half of the window. Move the slider marked DATA up and down until the LED window display reads “6 Super Mode” in the bottom half. Press the button marked “ENTER”. The LED should now display the words “Super Mode: off” in the top half of the window and “Select on/off” in the bottom half. Move the slider marked “DATA” up and down until the LED window displays “Super Mode: on” in the upper half of the window. Press the button marked “SETUP”, the LED display should now show “P00 -Untitled” in the upper half of the window.
- **Step 3 - Mixing Console** - On the RAMSA mixing console ensure all volume sliders are set at the bottom. Press the on/off switch located on the front panel upper right to the on position. The RAMSA mixer uses a Db scale for volume output, this means that a position of 0 is full volume, above 0 is a Db boost and below 0 is a Db reduction. Note: this does not refer to the physical position of the slider, but to the scale drawn on the console next to each slider. Move the sliders for channels 5 and 6 so that the black line in the center of the slider lines up with the position marked negative 10.

Move the sliders for channels 7 and 8 so that the black line in the center of the slider lines up with the position marked negative 5. Move the red sliders for the master volume control so that the white line in the center of the sliders is lined up with the position marked negative 10. Ensure the pan pot settings for channels 5 and 7 are set to A, knob all the way to the right. Ensure the pan pot settings for channels 6 and 8 are set to B, knob all the way to the left.

- **Step 4 - Ensoniq DP-4** - There are two of these processors located in the top two spaces of the audio rack. Turn the on/off switch for each unit to the on position. These take approximately 5 seconds to boot. Ensure volume settings for the bottom DP-4 are set with channel one and two (left two top and left two bottom) one notch mark past the halfway point. The right upper two and bottom two should be set to zero (full counterclockwise), as these two channels are not being used. Ensure all input and output volume settings for the bottom DP-4 are set to the halfway (12 o'clock) position. The marker for the volume control will face directly upward in this position.

- **Step 5 - RAMSA Subwoofer Processor** - Press the button marked "Power" on the front panel. A red light will be lit to indicate power is on.

- **Step 6 - Carver Power Amplifier** - Turn the switch marked "Power" to the on position. Ensure the volume settings for each channel are at maximum volume. This is when the volume controls are rotated fully in the clockwise direction.

- **Step 7 - RAMSA Power Amplifies** - Turn the switch marked "Power" to the on position for both RAMSA power amplifiers. These are located in the bottom spaces of the audio rack. Ensure that the volume is set to 10 (12 o'clock) for both the A and B channels of each of the two amplifiers. This will put the position indicators facing directly upward.

- **Step 8 - Execute Program** - The final step in bringing up the system is to start the software program. This procedure is detailed in the next section. Once the software is started, increase the slider marked "volume" on the EMAX II to the desired position. This slider will control the overall volume of the system. Use this slider to adjust overall volume up and down as desired, as it equally affects all subchannels on the EMAX II. Alteration of any of the other volume controls throughout the system will result in the speakers being thrown out of balance and severely degrade the localization capabilities of the system. To exit the program move mouse into graphic window and hit escape key. Turn off all audio equipment in reverse order.

B. SOFTWARE EXECUTION

The NPSNET-PAS software can currently be found the directory n/elsie/work3/roesli/NPSNET_SOUND. The executable is titled NPSPAS. Simply typing this command at the prompt will not properly start the program. In order to increase modularity and add

the ability to use the program with multiple terrains, there are a series of options that must be determined at run time.

1. Command line options

The following list provides a list of command line switches to set. This can also be obtained by typing NPSPAS -h at the command prompt. Following these switches is a list of the options available.

a. List of Options

- -h {for help}
- -i <broadcast interface> {to set local broadcast channel}
- -b <bank num> {to load midi bank}
- -c <config file> {to read config file}
- -m <machine name> {to choose master}
- -e <environment file> {to load environmentals}
- -x {to perform test}
- -d {to debug, no midi output}
- -w {no graphics window}
- -z <exercise> {exercise number}
- -p <port> {to set Network port}
- -g <group> {to set Multicast group}
- -t <ttl> {to set Multicast ttl}
- -n {to enable Multicast}

b. Usage

• -h: This simply prints the list of switches to the screen

• -i: Specifies which ethernet interface to use (there can be more than one per machine -- however, all our machines have exactly one. The name of the interface on the SGI Reality Engine equipped machines is “et0” and on all others is “ec0”).

• -b: This determines the bank number that the EMAX II will load upon execution. The default is bank 3, which is standard for all terrains currently being used by NPSNET-IV. The switch is invoked with a bank number as an argument. Example, -b 5, would load bank 5 upon execution.

• -c: This switch allows for different configuration files to be read upon execution. The configuration files available are: config.trg, config.benning, and config.hl. These configuration files contain the following data: name of the master or host machine, specify the use of round world coordinates, the exercise ID number, the environmental data file, and the network file. If any of these parameters are given by the another command line switch the config file parameters are overridden.

• -m: This determines which machine will be defined as the host entity. This is important, as the host position will act as the center of the sound world and all sounds generated will be based on this entity's position. The default host is "meatloaf". Example, -m gravy3, would make the user on gravy3 the host.

• -e: This switch allows the loading of the environmental data file. This provides the capability to load different geographic data for various environmental sound effects. Each terrain has many different properties and the environmental data is completely different. Example, -e environ_snd.dat, will load this file of geographic points with their associated sound data.

• -x: This will perform a test of the audio system by playing sounds individually through the forward right, forward left, right rear, and left rear speaker channels. This is provided for verifying setup when debugging changes to the program. If the sounds are heard in the correct order the directional algorithm can be assumed to be working correctly. This is also very handy to verify the external audio system when reconfiguring or setting up the hardware. It is very common to cross audio channels when setting up the system.

• -d: This will disable the transmission of MIDI data to the sampler for purposes of debugging program changes.

• -w: If run on a less capable machine this will prevent the graphic display window from being drawn. MIDI data output is not affected.

• -z: This is the DIS simulation exercise identifier and is required for the network code to read only the packets that apply to the selected exercise. This must be obtained from the user that initiates the simulation exercise.

• -p: Network port number.

• -g: Multicast group.

• -t: Multicast ttl. This determines the length of time a packet will stay alive on the internet and how far it will reach.

• -n: This switch will set up the network portion of the program to read packets using a multicast wrapper around the data packets being sent. This allows NPSNET-PAS and NPSNET-IV to be used over the internet.

c. Sample script file

Several script files have been developed to aid in command line execution. Here are a couple of script file examples script:

• File Name: demo-sound-benning-Ex7 Contents of file:

NPSPAS -n -p 3000 -t 3 -g 224.2.121.93 -e environ_snd.dat -c config.benning -z 7 $*

This script will launch the NPSNET-PAS sound server in multicast mode over network port 3000 with a ttl of 3 under group 224.2.121.93 using environmental file environ_snd.dat, configuration file config.benning and respond to data packets with an identification of exercise 7.

• File Name: demo-sound-trg Contents of file:

NPSPAS -c config/config.trg -m gravy5 -e environ_snd.dat $*

This script will launch the NPSNET-PAS sound server in broadcast mode using the config.trg configuration file, environ_snd.dat as the environmental file and assign gravy5 as the host.

Note the $* is used so that other options can be selected along with the script.

APPENDIX B: MIDI COMMANDS FOR NPSNET-PAS

This appendix contains a list and description of the MIDI commands used in NPSNET-PAS to control the EMAX II sampler [Ref. 21] and the Ensoniq DP-4 digital signal processors [Ref. 22].

A. MIDI commands for the EMAX II

1. MIDI note-on and note-off

The function *trigger_sound* is the primary function to send a MIDI note-on followed by a MIDI note-off command:

```
void trigger_sound(int volume, int sound, int midiport, int channel)
  {
  send_midi_command(midiport, (unsigned char) (NOTE_ON + channel));
  send_midi_command(midiport, (unsigned char) sound);
  send_midi_command(midiport, (unsigned char) volume);

  send_midi_command(midiport, (unsigned char) (NOTE_OFF + channel));
  send_midi_command(midiport, (unsigned char) sound);
  send_midi_command(midiport, (unsigned char) 0);
 }
```

The variable NOTE_ON is defined as 0x90 and the variable NOTE_OFF is defined as 0x80 (Table 1). Each command consists of three bytes. The first is the channel number, the second is the note to be played, and the third byte is the velocity. In MIDI, velocity refers to the amount of force applied to the keyboard when the note is played, which in turn corresponds to how loud the note will be sounded.

Function	Channel Number	Note Value	Velocity
Note On	0x90 thru 0x7f	0x00 thru 0x7f	0x00 thru 0x7f
Note Off	0x80 thru 0x7f	0x00 thru 0x7f	0x00 thru 0x7f

Table 1: MIDI note-on and note-off commands

2. Sequencer start and stop

Each bank on the EMAX II contains several sequences. The sequences serve two purposes. The first is to act as a file of MIDI data that can be played back in the form of a song. When the super mode function has been turned on, the sequence serves as the method of setting up the EMAX II to respond to any one of 16 MIDI channels and apply the data received on that channel to any preset chosen in the bank. This in effect is what makes the EMAX II a multitimbral machine. The bank set up for NPSNET-PAS contains four of these sequences. The first sequence is set up to receive the commands from *trigger_3D_sound* and also play the vehicle and environmental sound effects. The other sequences enable the playing of various status messages and a musical theme song when the program detects the lack of any entities on the network. The following two messages are the necessary bytes to control starting and stopping a MIDI sequence.

```
send_midi_command ( sr.midifd, (unsigned char)SONG_SELECT );
send_midi_command ( sr.midifd, (unsigned char)SONG_ON_MESSAGE);
send_midi_command ( sr.midifd, (unsigned char)START_SEQUENCE );

send_midi_command ( sr.midifd, (unsigned char)STOP_SEQUENCE );
```

The first byte informs the EMAX II that the next number will select a sequence. The second number indicates which sequence to load and the third byte is what starts the sequence playing. The fourth byte sent is the sequencer stop command, which will simply stop the music (Table 2).

Song Select Switch	Song Number Selected	Start Sequence	Start Sequence
0xf3	0x00 thru 0x31	0xfa	0xfc

Table 2: Commands for MIDI sequences

3. Loading a Bank

The EMAX II has several odd characteristics. One of them is the procedure for loading a bank [Ref. 23]. In order to load a bank via MIDI command, the user has to first preselect a receive channel for bank changes. This is done using the "MIDI" option under the "MASTER" menu selection. Then, to load the bank desired you first send a bogus preset change message on that channel, followed by the actual preset change message that corresponds to the bank to be loaded. If the bogus preset change message is not preceded by the proper bank preset message, the command will fail. In the graphics lab, the EMAX-II has been configured to respond to bank changes on MIDI channel 15. Thus the message for changing banks consists of four bytes, a preset change command on channel 15 followed by a bogus preset number, another preset change command on channel 15, and finally the number of the actual bank you wish to load (Table 3).

Preset Channel	Bogus Preset # Message	Preset Channel	Actual Bank Number to be Loaded
0xca	0z00 thru 0x63	0xca	0x63

Table 3: MIDI commands for loading a bank

4. MIDI Pitch Bend Command

Sending a pitch bend command will affect all sounds being generated on the assigned channel. The command consists of three bytes. The first byte informs the EMAX II that this is a pitch bend message. The least significant 14 bites of the next two bytes are used to determine the amount of increase or decrease in pitch (Table 4).

Status Byte	LSB	MSB
0xe0 thru 0xef	0x00 thru 0x7f	0s00 thru 0x7f

Table 4: MIDI commands for pitch bend

B. MIDI commands for the Ensoniq DP-4

The Ensoniq DP-4 parallel digital effects processors are extremely capable. They will respond to general MIDI specification program changes. In addition they will respond to a host of system exclusive MIDI commands (these are specific MIDI control messages designed by the manufacturer that are unique to a particular MIDI device or line of devices).

Currently the NPSNET-PAS sends standard MIDI program change messages to the processors. This simultaneously makes a preset change take effect for all 4 input channels (Table 5). The processors have been set up to receive this message on channel 5, however the machine can receive this information on any of the sixteen available midi channels.

Status Byte	Program selected
0xc0 thru 0xcf	0x00 thru 0x7f

Table 5. MIDI commands to change Ensoniq Preset

There are also other commands available using a system exclusive (SyxEx) header provided by the manufacturer [Ref. 22]. One example cited here is the virtual knob control. This command has the same effect as turning the main knob on the front panel of the DP-4. This allows program changes similar to those above. However, this command allows incrementing or decrementing the knob by a given amount vice programming a specific preset change. The message requires 11 bytes. The first four bytes are the message code. This informs the DP-4 that a SyxEx command is being sent. Byte 5 identifies the machine for which the command is intended. Byte 6 is the type of message being sent, e.g. command

message. Bytes 7 and 8 inform the DP-4 that the command will be applied to the Virtual Knob. Bytes 9 and 10 consist of either the two bytes corresponding to knob change up or knob change down. Notice in Table 6 both are listed, only two of these four bytes are sent. The second byte of the knob change up or down row is the number of turns the knob will make. The maximum is 99 or 0x63. The SysEx user guide for the DP-4 provides information on many more commands possible [Ref. 22]. One of the more interesting is the command that will change the effect algorithm for only one of the four processors at a time. This would allow a programmer to route four different effects to each of the four different output channels.

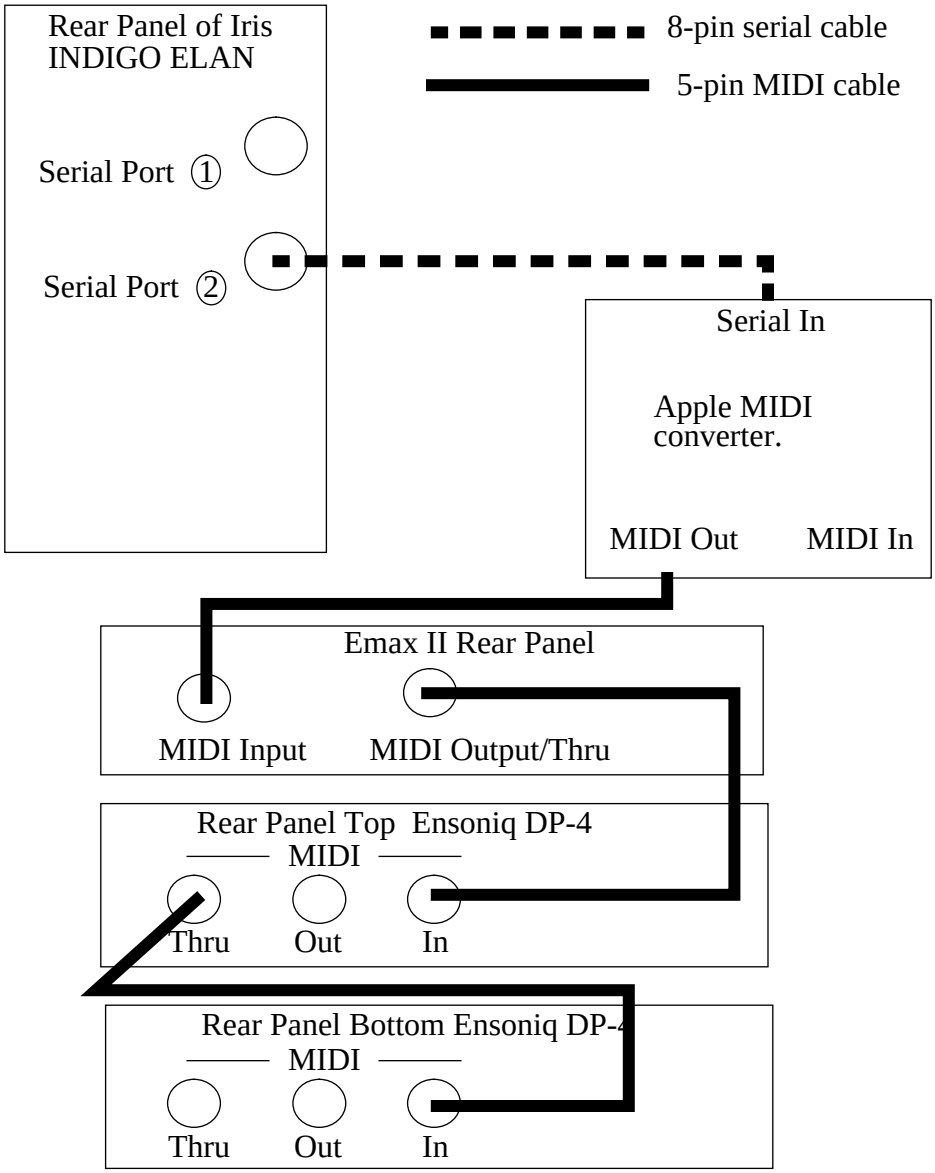
Message code	0xf0	0x0f	0x40	0x00
Machine ID	0x00			
Message	0x01			
Command Knob	0x00	0x03		
Knob Change Up	0x08	0x01		
Knob Change Down	0x00	0x01		
End Message	0xf7			

Table 6. System exclusive message for Virtual Knob

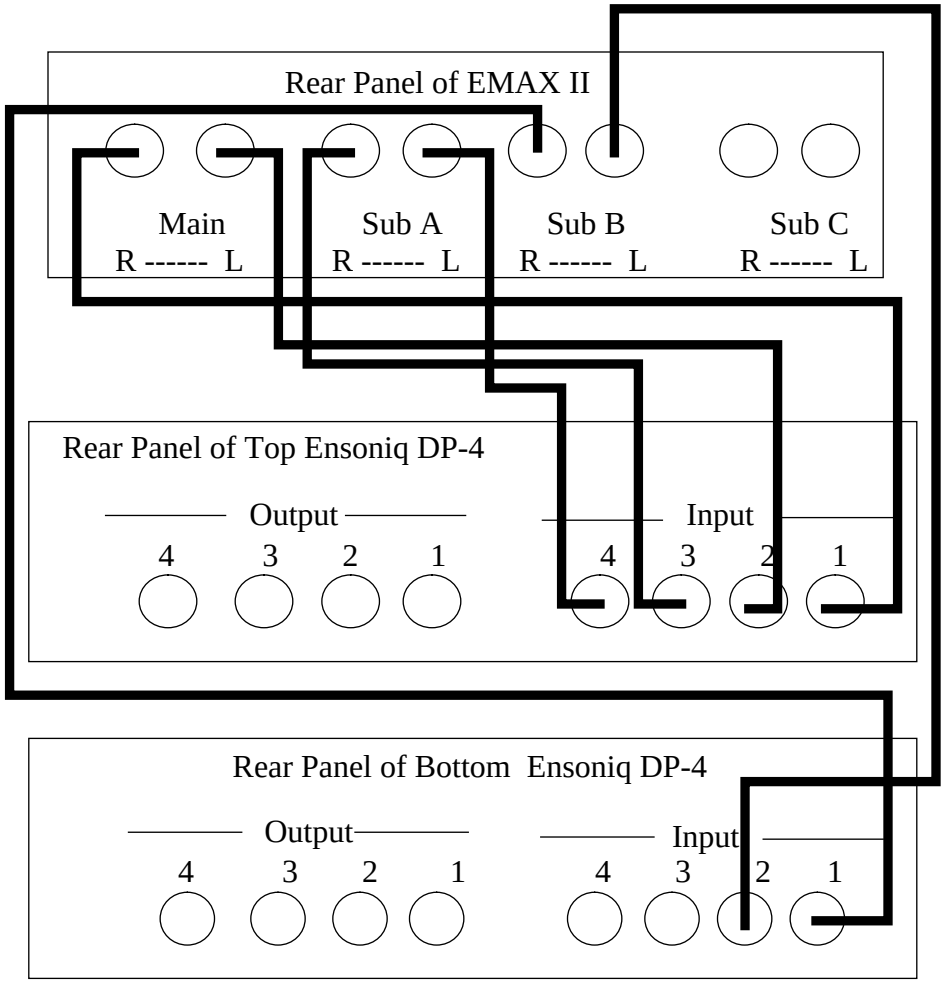
APPENDIX C: HARDWARE WIRING DIAGRAMS

This appendix serves as a supplement to Appendix A: Users Guide. It contains all information necessary to complete wiring of the hardware for the NPSNET-PAS system.

A. NPSNET-PAS MIDI CABLE CONNECTION

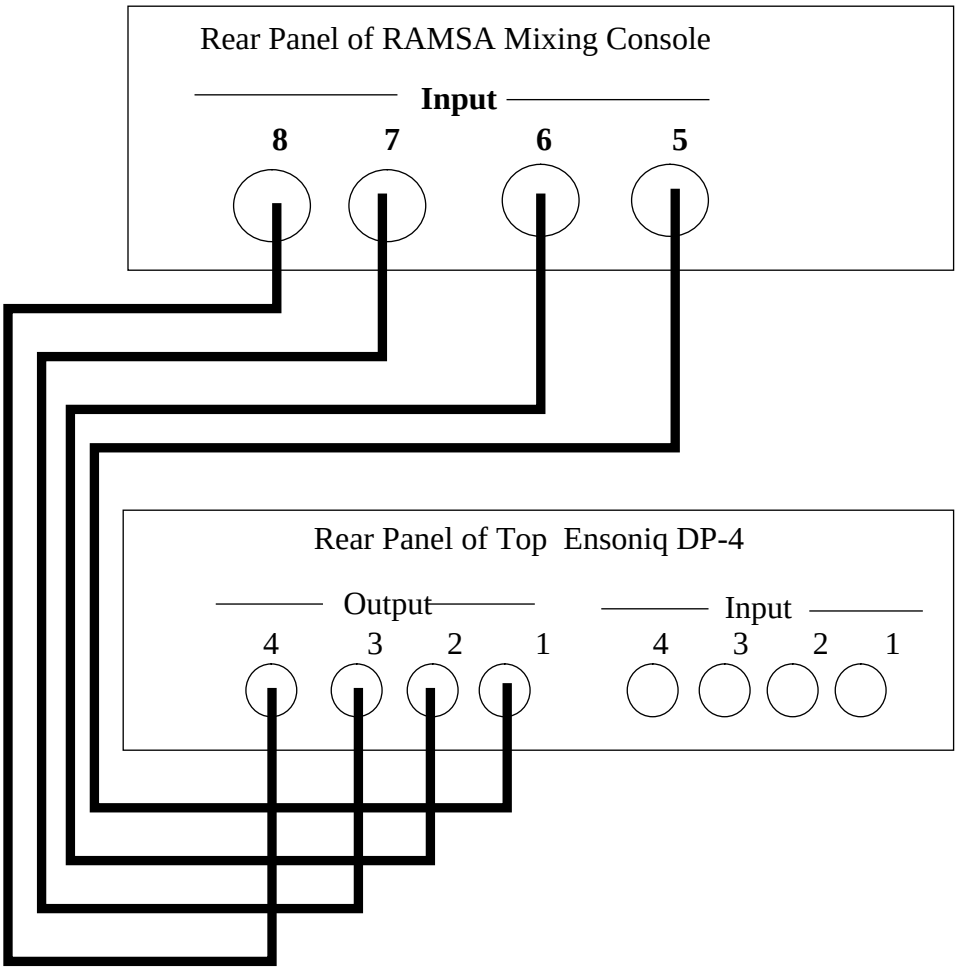


B. EMAX II TO ENSONIQ DP-4'S



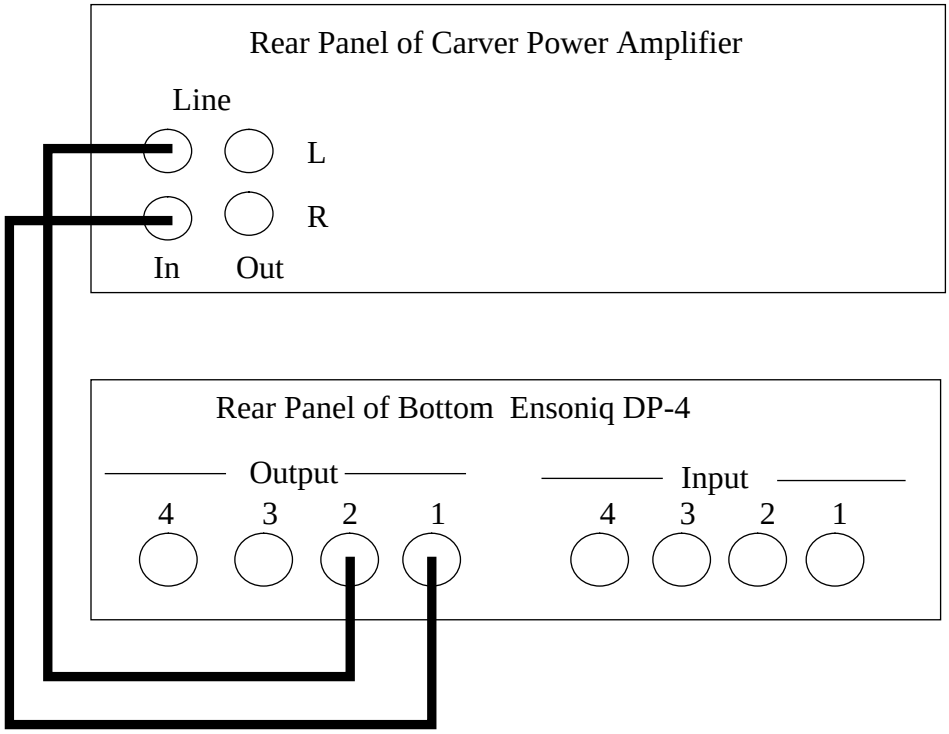
Connections are with 1/4" mono phone plug to 1/4"mono phone plug.

C. TOP DP-4 TO RAMSA MIXING CONSOLE



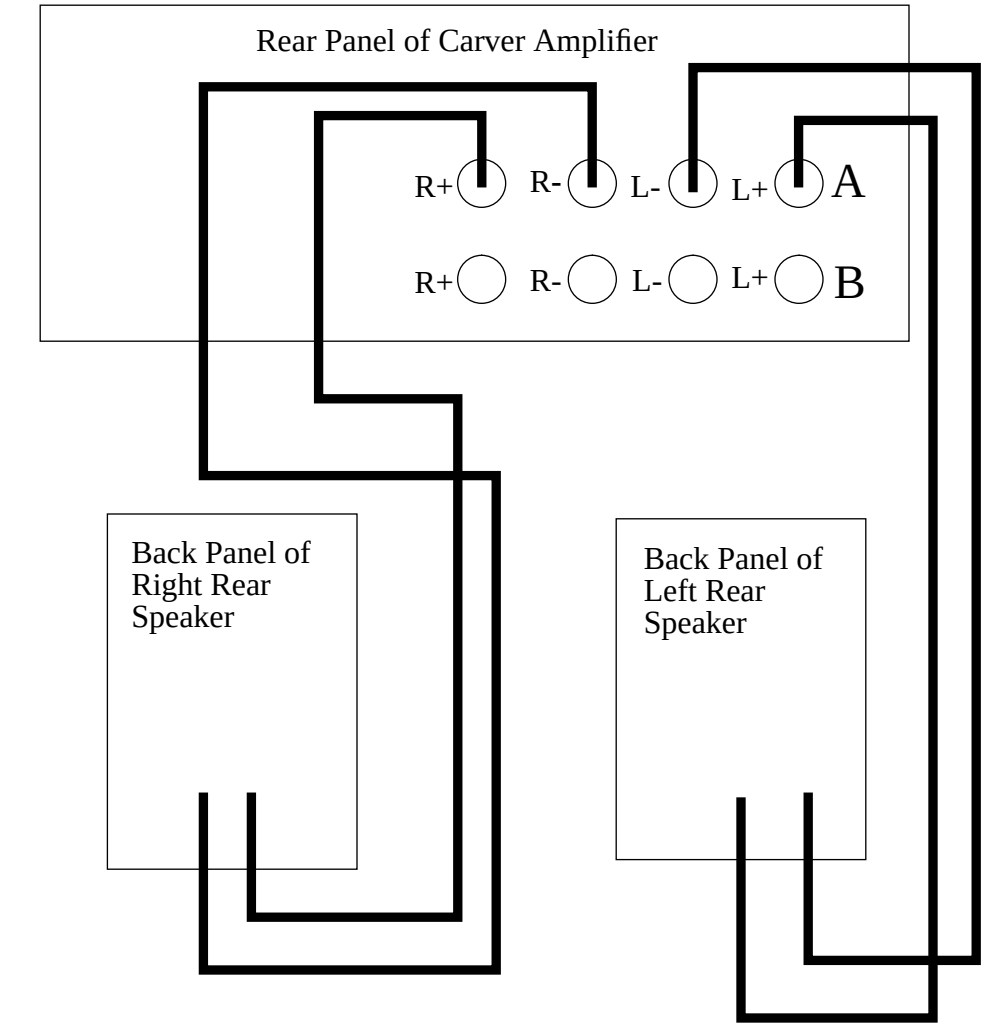
Cables are 1/4” mono phone plug for DP-4 to 3-pin XLR for mixing console

D. BOTTOM DP-4 TO CARVER AMPLIFIER



Cables are 1/4” mono phone plugs on DP-4 and RCA jacks going into the Carver amplfier.

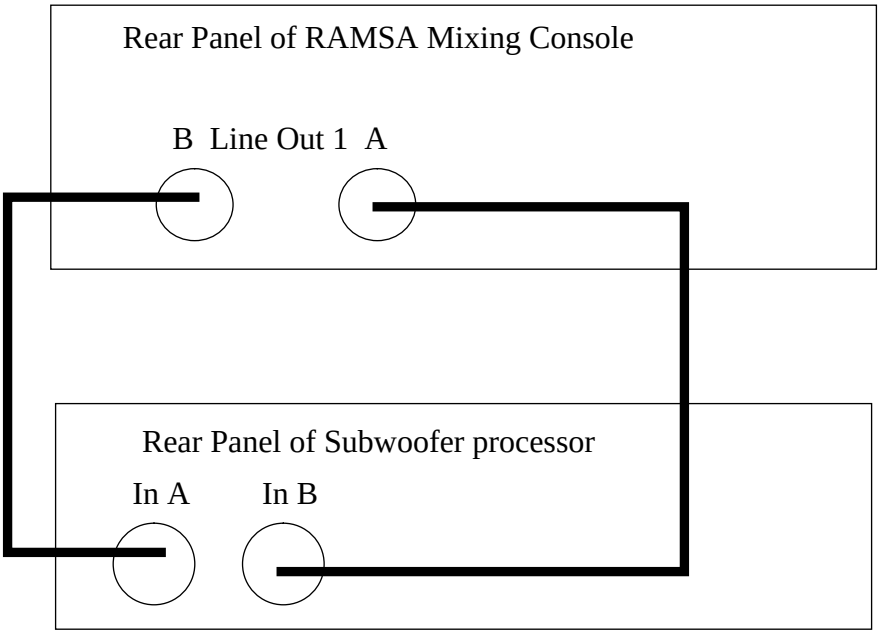
E. CARVER AMPLIFIER TO REAR SPEAKERS



Note : Polarity of speaker connections is required to match exactly, only left versus right.

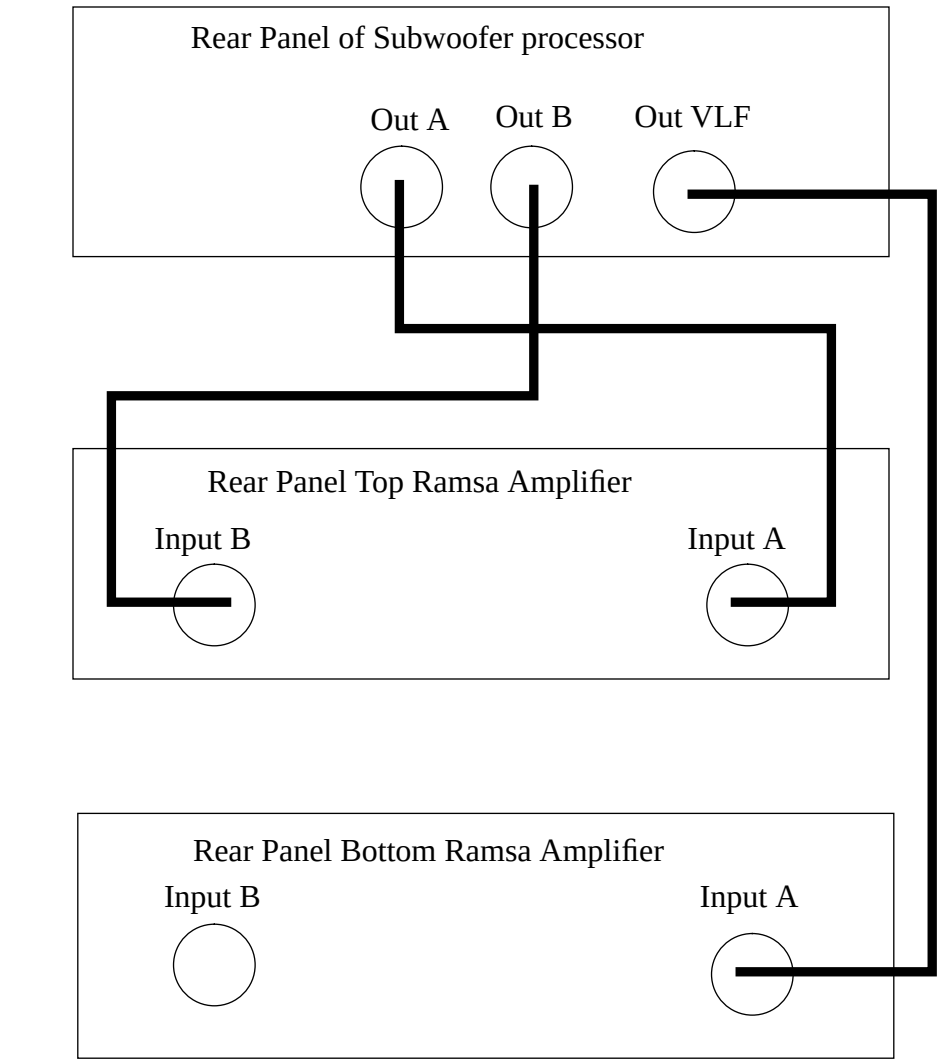
Wiring is 12 gauge speaker wire at both ends.

F. MIXING CONSOLE TO SUBWOOFER PROCESSOR



These cables have 3-pin XLR connectors on each end.

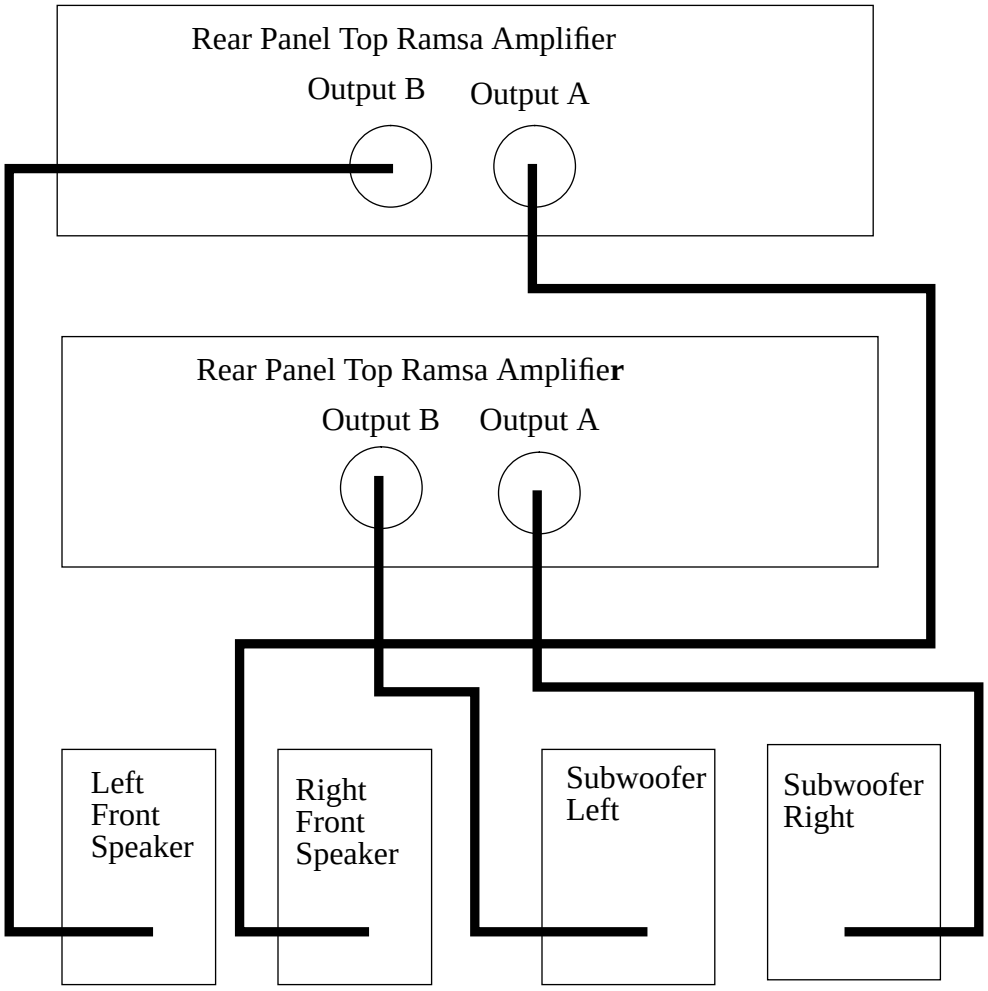
G. SUBWOOFER PROCESSOR TO RAMSA AMPLIFIERS



Cables are 1/4” mono phone plug at both ends

Note : The line running from Out VLF can go to either A or B input.

H. RAMSA AMPLIFIERS TO EXTERNAL SPEAKERS



Cables are 1/4” phone plug to 1/4” phone plug.

APPENDIX D: EMAX II CONFIGURATION

This appendix contains information regrading the set up procedures for the EMAX II and a detailed listing by table of the current NPSNET-PAS sound bank. Also, there are some helpful tips for manipulating data on the EMAX II.

A. SOUND BANK CONSTRUCTION

Learning to use the EMAX II can be an extremely time consuming undertaking. The user's manual is helpful in most situations, however a lot of the more complicated functionality of the sampler takes a lot of trial and error. It is half art and half science to set up the sampler and obtain the desired results. This section will provide a set of guidelines for programming the EMAX II. All of the functions cannot be listed. For additional information consult the EMAX II Owners Manuel [Ref. 21].

1. Sound Bank Operations

Let's begin by turning on the EMAX II. The first thing is to turn on the Syquest SCSI removable hard drive. This must be done before starting the EMAX II, otherwise the EMAX II will not recognize the drives. Once the Syquest drives have booted, turn on the EMAX II. The LED readout will show "P00 - Untitled". To load a sound bank press the button marked load bank. Move the DATA slider up and down until the bank to be loaded is shown in the lower half of the LED window. Press the ENTER button. It will take anywhere from a few to many seconds to load the sound bank, depending upon the size of the samples contained in the bank. Once a bank is loaded you can select a preset to play by moving the DATA slider up and down and, when the preset to be loaded is showing in the lower half of the LED window, press the ENTER button. The sampler will now play all of the sounds for that preset by pressing keys on the keyboard.

2. SAVING A SOUND BANK

For a first time user the best course of action is to load a bank and begin using all of the various functions. Do not worry about ruining the sound bank. If things get out of

hand just reload the bank. All of the changes made will be gone and the original state of the sound bank will be restored. If you make successful changes to the sound bank and wish to save them, press the button marked PRESET MANAGEMENT. Move the data slider up and down until you see “8 Save all 16 bit” in the LED window. Then press the ENTER button. The bank can either be written over or saved to another bank. The LED window will prompt the user for which course of action.

3. MULTI TIMBRAL MODE

Understanding the sequencer is perhaps the most important first step to understanding the construction of an NPSNET-PAS sound bank. The best way to understand this is to set up an imaginary sequence layout. The first step is to select a preset to be used as one of the instruments in the sequence and load that preset. Now select a sequence by pressing the SELECT button, move the DATA slider up and down until you find the sequence to record into. If you want to start a new sequence use the SETUP button to copy an existing sequence, then use the SETUP button again to delete the sequence just copied. This will remove all of the data, however the empty sequence will still be loaded as the current sequence. Now press the RECORD button and the PLAY button. When you begin to play the instrument the sequencer will begin recording input from the keyboard. Press the STOP button. The actions applied to the keyboard have now been recorded on track one of the sequence. This means that if the SUPERMODE has been selected and this sequence is loaded the sampler will apply MIDI data on channel one to this preset. Press the SETUP button and select the option TRACK STATUS. The display will show sixteen slots corresponding to the sixteen channels. Using the arrow keys move the flashing cursor under the second channel and press the ON/YES or OFF/NO buttons to change the status of track two to record. An R will be displayed on track two when this is done. Now press the setup button to return to the preset menu in the LED window. Select the instrument you wish to place on track 2. Press the RECORD and then the PLAY button. Play the keyboard for a few seconds and then press the STOP button. The instrument played has now been assigned to track 2. This procedure can continue on until all sixteen channels have been assigned a preset. If the preset were to contain sound effects instead of samples of a

particular musical instrument, the method behind NPSNET-PAS begins to become clear. Basically four copies of the same preset have been assigned to four different MIDI channels. Each preset has been assigned a specific output channel and panned either to the left or to the right.

4. DYNAMIC PROCESSING

This is where the presets can be manipulated for NPSNET-PAS. For example, press the DYNAMIC PROCESSING button. Hit the lowest key to be affected by the changes that need to be made and press the ENTER button, now hit the highest key to be affected by the changes and press the ENTER button. Move the data slider up and down and select KEYBOARD MODE by pressing the ENTER button when this is displayed in the LED window. This is where you can decide which subchannel to assign for the preset. There are four subchannels available. When the one desired is selected press the ENTER button. Now move the data slider up and down until PANNING is shown in the LED window. Press the ENTER button and the LED window will display two arrows in the center bottom half of the LED window. Move the DATA slider up and down to pan the preset left or right as desired and press the ENTER button. Do not forget at this point that if changes have been made, you have to save the data, otherwise the next bank load will erase all of the work that has been done. A habit of saving every five or ten minutes prevents a lot of headaches.

In addition to the information provided above there is the necessity to initially set up the sound bank. This involves moving voices and preset around and is beyond the scope of this appendix. Consult the EMAX II Owners Manuel for instruction on this procedure. Additional help can be obtained by calling the technical assistants at E-mu Corporation located in Scotts Valley, California. The following tables list the current preset and sequence configuration of the NPSNET-PAS sound bank. At the time of this writing the bank was number three and the name of the bank was “NPSNET HL”.

B. TABLES OF SOUND BANK LAYOUT

Sample	Note Value	Hex Value	Output Channel	Pan Setting
Rifle	C-2	0x30	SubA	Right
Rifle Large	D-2	0x32	SubA	Right
Rile-Auto	E-2	0x34	SubA	Right
M-60	F-2	0x35	SubA	Right
25mm	G-2	0x37	SubA	Right
Explosion1	A-2	0x39	SubA	Right
Explosion2	B-2	0x3B	SubA	Right
Explosion3	C-3	0x3C	SubA	Right
Exposion4	D-3	0x3E	SubA	Right
Explosion5	E-4	0x40	SubA	Right
Explosion6	F-4	0x41	SubA	Right
Sm. Missile	G-4	0x43	SubA	Right
Med.Missile	A-4	0x45	SubA	Right
Lg. Missile	B-4	0x47	SubA	Right
Cannon1	C-5	0x48	SubA	Right
Cannon2	D-5	0x4A	SubA	Right
Lg.Artillery	E-5	0x4C	SubA	Right
M1 Fire	F-5	0x4D	SubA	Right
Seagulls	G-5	0x4F	SubA	Right
Crickets	A-5	0x51	SubA	Right

Table 1: Preset 01 MIDI CH 1

Sample	Note Value	Hex Value	Output Channel	Pan Setting
Rifle	C-2	0x30	SubA	Left
Rifle Large	D-2	0x32	SubA	Left
Rile-Auto	E-2	0x34	SubA	Left
M-60	F-2	0x35	SubA	Left
25mm	G-2	0x37	SubA	Left
Explosion1	A-2	0x39	SubA	Left
Explosion2	B-2	0x3B	SubA	Left
Explosion3	C-3	0x3C	SubA	Left
Exposion4	D-3	0x3E	SubA	Left
Explosion5	E-4	0x40	SubA	Left
Explosion6	F-4	0x41	SubA	Left
Sm. Missile	G-4	0x43	SubA	Left
Med. Missile	A-4	0x45	SubA	Left
Lg. Missile	B-4	0x47	SubA	Left
Cannon1	C-5	0x48	SubA	Left
Cannon2	D-5	0x4A	SubA	Left
Lg.Artillery	E-5	0x4C	SubA	Left
M1 Fire	F-5	0x4D	SubA	Left
Seagulls	G-5	0x4F	SubA	Left
Crickets	A-5	0x51	SubA	Left

Table 2: Preset 02 MIDI channel 2

Sample	Note Value	Hex Value	Output Channel	Pan Setting
Rifle	C-2	0x30	SubB	Right
Rifle Large	D-2	0x32	SubB	Right
Rile-Auto	E-2	0x34	SubB	Right
M-60	F-2	0x35	SubB	Right
25mm	G-2	0x37	SubB	Right
Explosion1	A-2	0x39	SubB	Right
Explosion2	B-2	0x3B	SubB	Right
Explosion3	C-3	0x3C	SubB	Right
Exposion4	D-3	0x3E	SubB	Right
Explosion5	E-4	0x40	SubB	Right
Explosion6	F-4	0x41	SubB	Right
Sm. Missile	G-4	0x43	SubB	Right
Med. Missile	A-4	0x45	SubB	Right
Lg. Missile	B-4	0x47	SubB	Right
Cannon1	C-5	0x48	SubB	Right
Cannon2	D-5	0x4A	SubB	Right
Lg.Artillery	E-5	0x4C	SubB	Right
M1 Fire	F-5	0x4D	SubB	Right
Seagulls	G-5	0x4F	SubB	Right
Crickets	A-5	0x51	SubB	Right

Table 3: Preset 03 MIDI channel 4

Sample	Note Value	Hex Value	Output Channel	Pan Setting
Rifle	C-2	0x30	SubB	Left
Rifle Large	D-2	0x32	SubB	Left
Rile-Auto	E-2	0x34	SubB	Left
M-60	F-2	0x35	SubB	Left
25mm	G-2	0x37	SubB	Left
Explosion1	A-2	0x39	SubB	Left
Explosion2	B-2	0x3B	SubB	Left
Explosion3	C-3	0x3C	SubB	Left
Exposion4	D-3	0x3E	SubB	Left
Explosion5	E-4	0x40	SubB	Left
Explosion6	F-4	0x41	SubB	Left
Sm. Missile	G-4	0x43	SubB	Left
Med. Missile	A-4	0x45	SubB	Left
Lg. Missile	B-4	0x47	SubB	Left
Cannon1	C-5	0x48	SubB	Left
Cannon2	D-5	0x4A	SubB	Left
Lg.Artillery	E-5	0x4C	SubB	Left
M1 Fire	F-5	0x4D	SubB	Left
Seagulls	G-5	0x4F	SubB	Left
Crickets	A-5	0x51	SubB	Left

Table 4: Preset 4 MIDI channel 5

Sample	Note Value	Hex Value	Output Channel	Pan Setting
Jet Rumble	G-1	0x2B	Main	Center
Jet Rumble	A-1	0x2D	Main	Center
Jet Rumble	B-1	0x2F	Main	Center
Jet Rumble	C-2	0x30	Main	Center

Table 5: Preset 06 - Vehicles

Sample	Note Value	Hex Value	Output Channel	Pan Setting
Helicopter	G-1	0x2B	Main	Center
Default Vehicle	A-1	0x2D	Main	Center
Jet Rumble	B-1	0x2F	Main	Center
M1A1 Tank	C-2	0x30	Main	Center

Table 6: Preset 07 - Vehicles

Preset assigned	Channel	Status
Preset 01	1	P
Preset 02	2	P
Preset 04	4	P
Preset 05	5	P
Preset 06	7	P
Preset 08	8	P

Table 7. Sequence 00 SFX

Preset assigned	Channel	Status
Preset 09	1	P
Preset 10	2	P
Preset 11	3	P
Preset 11	4	P
Preset 12	5	P

Table 8. Sequence 01 Theme

Preset assigned	Channel	Status
Preset 14	1	P

Table 9. Sequence 02 Activated

Preset assigned	Channel	Status
Preset 14	1	P

Table 10. Sequence 03 DEACTIV.

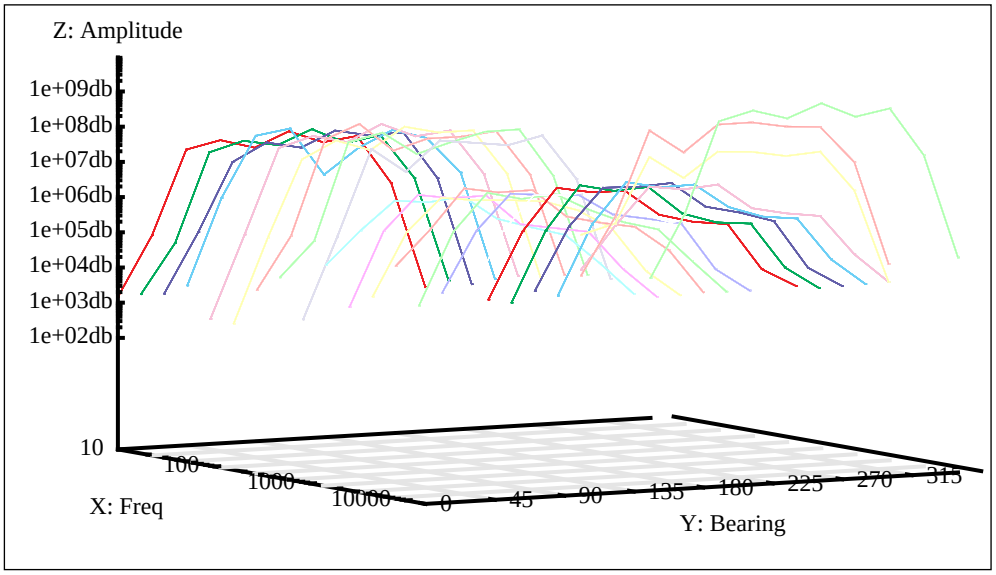
Preset assigned	Channel	Status
Preset 01	1	P

Table 11. Vehicle Destroyed

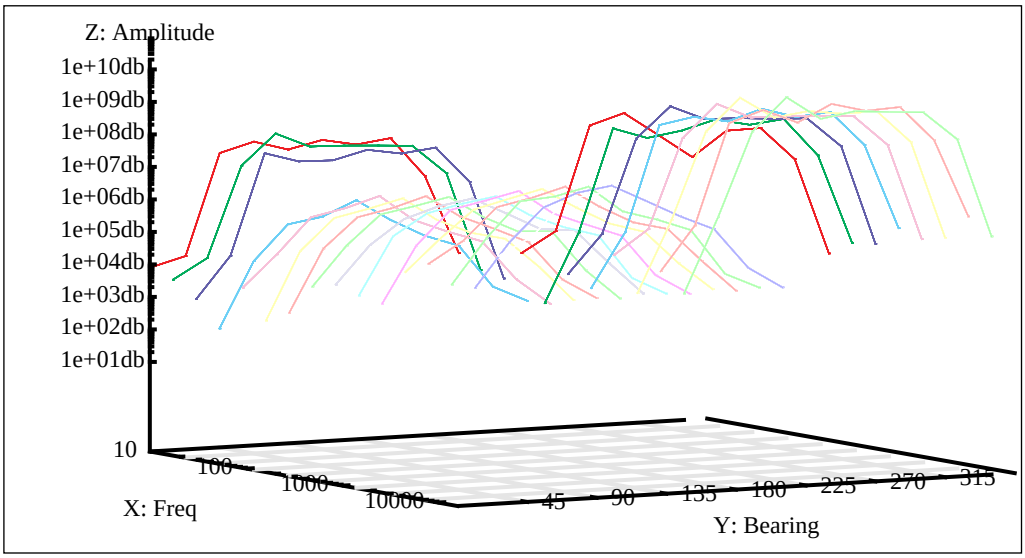
APPENDIX E: ACOUSTIC ANALYSIS GRAPHS

This appendix contains graphs generated using the Hewlett Packard Spectral Analyzer and Unix GNU Plot software.

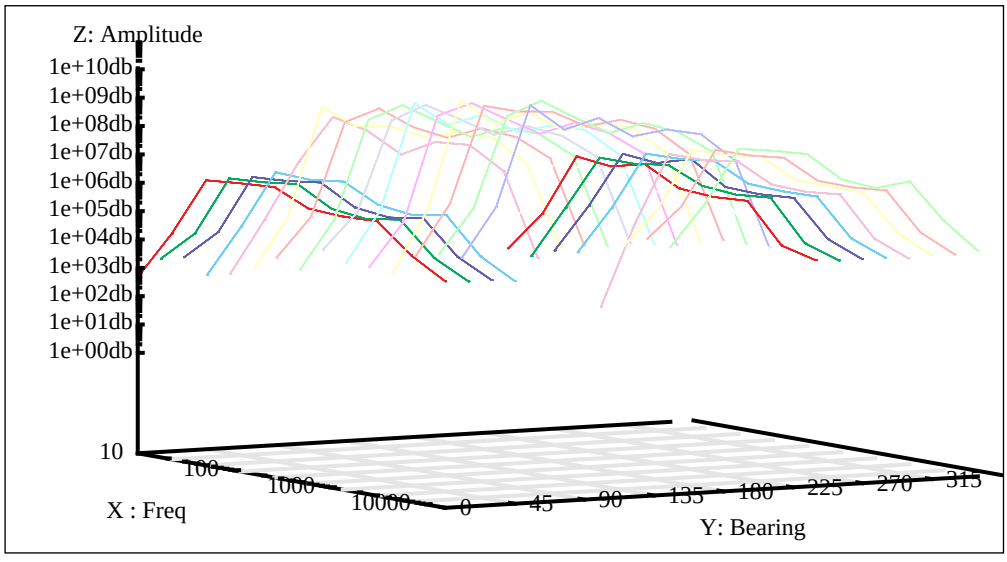
A. Three Dimensional Plot: Forward Right



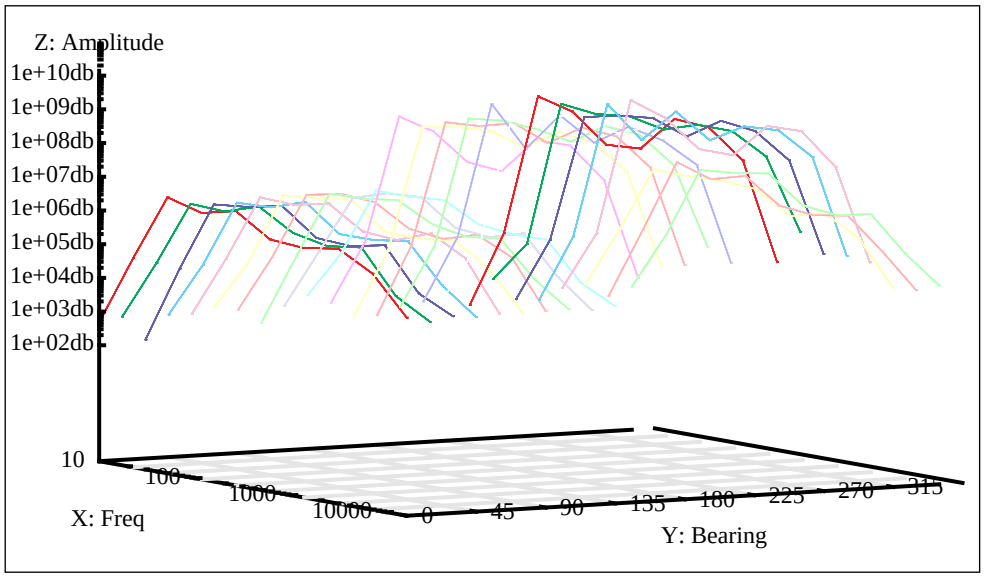
B. Three Dimensional Plot: Forward Left



C. Three Dimensional Plot: Right Rear



D. Three Dimensional Plot: Left Rear



LIST OF REFERENCES

1 Cherry, E. C., “Some Experiments on the Recognition of Speech With One or Two Ears”, J. Acoust. Soc. Am., pp. 22, 61-62, 1953.

2 Lord Rayleigh, Strutt, J.W., “On Our Perception of Sound Direction”, Phil. Mag., pp. 13, 214-232, 1907.

3 Wenzel, E. M., “Localization in Virtual Acoustic Displays”, Presence, Vol. 1, No. 1, pp.80, Winter 1992.

4 Gardner, M. B., Gardner, R. S. “Problem of Localization in the Median Plane: Effect of Pinnae Cavity Occlusion”, J. Acoust. Soc. Am., pp. 53, 400-408, 1973.

5 Plenge, G., “On the Differences Between Localization and Lateralization”, Journal Acoustic Soc. Am., pp. 56, 944-951, 1974.

6 Wenzel, E. M., Begault, D. R., “Techniques and Applications for Binaural Sound Manipulation in Human-Machine Interfaces”, Ames Research Center, Moffett Field, CA, 1990.

7 Blauert, J., “Spatial Hearing: The Psychophysics of Human Sound Localization”, MIT Press: Cambridge, MA, 1983.

8 Parker, S. P. A., Oldfield, S. R., “Acuity of Sound Localization: A Topography of Auditory Space. II. Pinna Cues Absent”, Perception, pp. 13, 601 - 617, 1984.

9 Mills, A. W., “Auditory Localization, Foundations of Modern Auditory Theory”, Vol. II, Academic: New York, NY, 1972.

10 O’Donnell, Bob, “What is MIDI, Anyway?”, Electronic Musician, pp.74, January 1991.

11 International MIDI Association, “1.0 MIDI Specification”, copyright 1983.

12 Burgess, David A., “Real-Time Audio Spatialization with Inexpensive Hardware”, Georgia Institute of Technology, October 1992.

13 Wilkinson, Scott; “The Thrill of Adventure”, Electronic Musician, pp. 22, June 1993.

14 Evans, Brian; “Enhancing Scientific Animations with Sonic Maps, An Introduction to Data Sonification”, Course 81 Notes, pp. 3.5 - 3.12, ACM SIGGRAPH `93.

15 Zyda, M., Pratt, D., Falby, J., Barham, P., Kelleher, K., “NPSNET and the Naval Postgraduate School Graphics and Video Laboratory”, Presence, Vol. 2, No. 3, Summer 1993.

16 Martens, William; “Demystifying Spatial Audio”, Ono-Sendai Corporation, 1992.

17 Taghavy, Darius, “Take a Roller Coaster Ride in Rolands’s Sound Space”, Electronic Musician, August 1994.

18 Institute for Simulation and Training, “Standard for Information Technology - Protocols for Distributed Interactive Simulation Applications”, Version 2.0 Third Draft, IST-PD-90-2, Orlando, FL, May 1993.

19 Begault, Durand R., “Preferred Sound Intensity Increase for Sensation of Half Distance”, Perceptual and Motor Skills, pp. 72, 1019-1029, 1991.

20 Ballou, Glen, “Handbook for Sound Engineers”, Howard W. Sams & Company, 1991.

21 E-mu Systems, Incorporated, “EMAX II 16-bit Digital Sound System Operation Manual”, E-mu Systems, Incorporated, 1989.

22 Ensoniq Corporation, “DP/4 MIDI SysEx Implementation Specification version 1.0”, 1992.

23 E-mu Systems, Incorporated, “EMAX II and RS-422 Specs”, Software Revision 1.0,2.0, Models, 1989.

INITIAL DISTRIBUTION LIST

1	Defense Technical Information Center Cameron Station Alexandria, VA 22304-6145	2
2	Dudley Knox Library Code 052 Naval Postgraduate School Monterey, CA 93943-5002	2
3	Chairman, Code CS Computer Science Department Naval Postgraduate School Monterey, CA 93943	2
4	Dr Michael M. Zyda, Code CS/ZK Computer Science Department Naval Postgraduate School Monterey, CA 93943	10
5	Dr David R. Pratt, Code CS/PR Computer Science Department Naval Postgraduate School Monterey, CA 93943	2
6	Dr Sheng Kwak, Code CS Computer Science Department Naval Postgraduate School Monterey, CA 93943	2
7	John Falby, Code CS Computer Science Department Naval Postgraduate School Monterey, CA 93943	2
8	Paul Barham, Code CS Computer Science Department Naval Postgraduate School Monterey, CA 93943	1

9 Susannah Bloch, Code CS 1
Computer Science Department
Naval Postgraduate School
Monterey, CA 93943